Deep Unsupervised Feature Learning for Natural Language Processing

Stephan Gouws
MIH Media Lab, Stellenbosch University
Stellenbosch, South Africa
stephan@ml.sun.ac.za

Abstract

Statistical natural language processing (NLP) builds models of language based on statistical features extracted from the input text. We investigate deep learning methods for unsupervised feature learning for NLP tasks. Recent results indicate that features learned using deep learning methods are not a silver bullet and do not always lead to improved results. In this work we hypothesise that this is the result of a disjoint training protocol which results in mismatched word representations and classifiers. We also hypothesise that modelling long-range dependencies in the input and (separately) in the output layers would further improve performance. We suggest methods for overcoming these limitations, which will form part of our final thesis work.

1 Introduction

Natural language processing (NLP) can be seen as building models $h : \mathcal{X} \rightarrow \mathcal{Y}$ for mapping an input encoding $x \in \mathcal{X}$ representing a natural language (NL) fragment, to an output encoding $y \in \mathcal{Y}$ representing some construct or formalism used in the particular NLP task of interest, e.g. part-of-speech (POS) tags, begin-, inside-, outside (BIO) tags for information extraction, semantic role labels, etc.

Since the 90s, the predominant approach has been statistical NLP, where one models the problem as learning a predictive function h for mapping from $h : \mathcal{X} \rightarrow \mathcal{Y}$ using machine learning techniques. Machine learning consists of a hypothesis function which learns this mapping based on latent or explicit features extracted from the input data.

In this framework, h is usually trained in a supervised setting from labelled training pairs $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. Additionally, the discriminant function h typically operates on a transformed representation of the data to a common feature space encoded as a *feature vector* $\phi(x)$, and then learns a mapping from feature space to the output space, $h : \phi(x) \rightarrow y$. In supervised learning, the idea is generally that features represent strong discriminating characteristics of the problem gained through manual engineering and domain-specific insight.

$\vec{x} \in \mathcal{X}$	**the**	**cat**	**sits**	**on**	**the**	**mat**
$\phi(\vec{x})$	$\phi(x_1)$	$\phi(x_2)$	$\phi(x_3)$	$\phi(x_4)$	$\phi(x_5)$	$\phi(x_6)$
$\vec{y} \in \mathcal{Y}$	B-NP	I-NP	B-VP	O	B-NP	I-NP
NE Tags	**[the cat]$_{\text{NP}}$**		**[sits]$_{\text{VP}}$**	**[on]$_{\text{O}}$**	**[the mat]$_{\text{NP}}$**	

Table 1: Example NLP syntactic chunking task for the sentence "the cat sits on the mat". $\mathcal{X}$ represents the words in the input space, $\mathcal{Y}$ represents labels in the output space. $\phi(\vec{x})$ is a feature representation for the input text $\vec{x}$ and the bottom row represents the output named entity tags in a more standard form.

As a concrete example, consider the task of syntactic chunking, also called "shallow parsing", (Gildea and Jurafsky, 2002): Given an input string, e.g.

"the cat sits on the mat",

the chunking problem consists of labelling segments of a sentence with syntactic constituents such as noun or verb phrases (NPs or VPs). Each word is assigned one unique tag often encoded using the BIO encoding[1]. We represent the input text as a vector of words $x_i \in \vec{x}$, and each word's corresponding label is represented by $y_i \in \vec{y}$ (see Table 1). Given a feature generating function $\phi(x_i)$ and a set of labelled training pairs $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, the task then reduces to learning a suitable mapping $h : \phi(\mathcal{X}) \rightarrow \mathcal{Y}$.

Most previous works have focused on manually engineered features and simpler, linear models, including "shallow" model architectures, like the perceptron (Rosenblatt, 1957), linear SVM (Cortes and Vapnik, 1995) and linear-chain conditional random fields (CRFs) (Lafferty, 2001). However, a shallow learning architecture is only as good as its input features. Due to the complex nature of NL, deeper architectures may be re-

[1]E.g. B-NP means "begin NP", I-NP means "inside NP", and O means other/outside.

Proceedings of the NAACL HLT 2012 Student Research Workshop, pages 48–53,
Montréal, Canada, June 3-8, 2012. ©2012 Association for Computational Linguistics

quired to learn data representations which contain the appropriate level of information for the task at hand. Prior to 2006, it was computationally infeasible to perform inference in hierarchical ("deep"), non-linear models such as multi-layer perceptrons with more than one hidden layer. However, Hinton (2006) proposed an efficient, layer-wise greedy method for learning the model parameters in these architectures, which spurred a renewed interest in deep learning research.

Still, creating annotated training data is labour-intensive and costly, and manually designing and extracting discriminating features from the data to be used in the learning process is a costly procedure requiring significant levels of domain expertise. Over the last two decades, the growth of available unlabeled data $x \in \mathcal{X}$ and the ubiquity of scalable computing power has shifted research focus to unsupervised approaches for automatically *learning* appropriate feature representations $\phi(x)$ from large collections of unlabeled text.

Several methods have been proposed for unsupervised feature learning, including simple k-means clustering (Lloyd, 1982), Brown clustering (Brown et al., 1992), mutual information (Shannon and Weaver, 1962), principal components analysis (PCA) (Jolliffe, 2002), and independent component analysis (ICA) (Hyvärinen et al., 2001).

However, natural language has complex mappings from text to meaning, arguably involving higher-order correlations between words which these simpler methods struggle to model adequately. Advances in the "deep learning" community allow us to perform efficient unsupervised feature learning in highly complex and high-dimensional input feature spaces, making it an attractive method for learning features in e.g. vision or language (Bengio, 2009).

The standard deep learning approach is to learn lower-dimensional *embeddings* from the raw high-dimensional[2] input space $\mathcal{X}$ to lower dimensional (e.g. 50-dimensional) feature spaces in an unsupervised manner, via repeated, layer-wise, non-linear transformation of the input features, e.g.

$$\hat{y} = f^{(k)}(\cdots f^{(2)}(f^{(1)}(\vec{x}))\cdots),$$

where $f^{(i)}(x)$ is some non-linear function (typically tanh) for which the parameters are learned by back propagating error gradients. This configuration is referred to as a "deep" architecture with k layers (see Figure 1 for an example).

For feature generation, we present a trained network with a new vector $\vec{x}$ representing the input data on its input layer. After performing one iteration of forward-propagation through the network, we can then view the activation values in the hidden layers as dense, so-called "distributed representations" (features) of the input data. These features can in turn be passed to an output classifier layer to produce some tagging task of interest. Recent work in deep learning show state-of-the-art results in part-of-speech parsing, chunking and named-entity tagging (Collobert, 2011), however performance in more complex NLP tasks like entity and event disambiguation and semantic role labelling are still trailing behind.

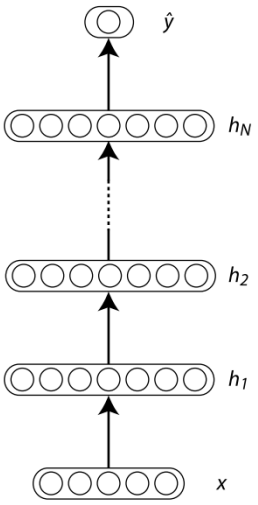

Figure 1: Example of a deep model. The input vector x is transformed into the hidden representation, here denoted as h_1, using an affine transformation W and a non-linearity. Each subsequent hidden layer h_k takes as input the output of its preceding layer $h_{(k-1)}$ (Bengio, 2009).

In this work we focus specifically on extending current state of the art deep neural models to improve their performance on these more difficult tasks. In the following section we briefly review and discuss the merits and limitations of three of the current state of the art deep learning models for NLP. We then identify our primary research questions and introduce our proposed future work roadmap.

2 Current State-of-the-Art and Limitations

Most work builds on the idea of a neural probabilistic language model (NPLM) where words are represented by learned real-valued embeddings, and a neural network combines word embeddings to predict the most likely next word. The first successful NPLM was introduced by Bengio et al. in 2003 (Bengio et al., 2003).

[2]E.g. a "one-hot" 50,000-dimensional vector of input words, with a '1' indicating the presence of the word at that index, and a '0' everywhere else.

Historically, training and testing these models were slow, scaling linearly in the vocabulary size. However, several recent approaches have been proposed which overcome these limitations (Morin and Bengio, 2005), including the work by Collobert and Weston (2008) and Mnih and Hinton (2009) discussed next.

2.1 Collobert & Weston (2008)

Collobert and Weston (2008) present a discriminative, non-probabilistic, non-linear neural language model that can be scaled to train over billions of words since each training iteration only computes a loss gradient over a small stochastic sample of the training data.

All K-dimensional word embeddings are initially set to a random state. During each training iteration, an n-gram is read from the training data and each word is mapped to its respective embedding. All embeddings are then concatenated to form a nK-length positive training vector. A corrupted n-gram is also created by replacing the n'th (last) word by some word uniformly chosen from the vocabulary. The training criterion is that the network must predict positive training vectors with a score at least some margin higher than the score predicted for corrupted n-grams. Model parameters are trained *simultaneously* with word embeddings via gradient descent.

2.2 The Hierarchical Log Bilinear (HLBL) Model

Mnih and Hinton (2007) proposed a simple probabilistic linear neural language model called the log bilinear (LBL) model. For an n-gram context window, the LBL model concatenates the first $(n-1)$ K-dimensional word embeddings and then learns a linear mapping from $\mathbb{R}^{(n-1)K}$ to $\mathbb{R}^K$ for predicting the embedding of the nth word. For predicting the next word, the model outputs a probability distribution over the entire vocabulary by computing the dot product between the predicted embedding and the embedding for each word in the vocabulary in an output softmax layer. This softmax computation is linear in the length of the vocabulary for each prediction, and is therefore the performance bottleneck.

In follow-up work, Mnih and Hinton (2009) speed up training and testing time by extending the LBL model to predict the next word by hierarchically decomposing the search through the vocabulary by traversing a binary tree constructed over the vocabulary. This speeds up training and testing exponentially, but initially reduces model performance as the construction of the binary partitioning has a strong effect on the model's performance. They introduce a method for bootstrapping the construction of the tree by initially using a random binary tree to learn word embeddings, and then rebuilding the tree based on a clustering of the learned embeddings. Their final results are superior to the standard LBL in both model perplexity and model testing time.

2.3 Recursive Neural Networks (RNNs)

Socher (2010; 2011) introduces a recursive neural network (RNN) framework for parsing natural language. Previous approaches dealt with variable-length sentences by either

i using a window approach (shifting a window of n words over the input, processing each fixed-size window at a time), or

ii by using a convolutional layer where each word is convolved with its neighbours within some sentence- or window-boundary.

RNNs operate by recursively applying the same neural network to segments of its input, thereby allowing RNNs to naturally operate on variable-length inputs. Each pair of neighbouring words is scored by the network to reflect how likely these two words are considered to form part of a phrase. Each such operation takes two K-dimensional word vectors and outputs another K-dimensional vector and a score. Socher (2010) proposes several strategies (ranging from local and greedy to global and optimal) for choosing which pairs of words to collapse into a new K-dimensional vector representing the phrase comprised of the two words. By viewing these collapsing operations as branch merging decisions, one can construct a binary parse tree over the words in a bottom-up fashion.

2.4 Discussion of Limitations

Neural language models are appealing since they can more easily deal with missing data (unknown word combinations) due to their inherent continuous-space representation, whereas n-gram language models (Manning et al., 1999) need to employ (sometimes ad hoc) methods for *smoothing* unseen and hence zero probability word combinations.

The original NPLM performs well in terms of model perplexity on held-out data; however, its training and testing time is very slow. Furthermore, it provides no support for handling multiple word senses, the property that any word can have more than one meaning, since each word is assigned an embedding based on its literal string representation (i.e. from a lookup table).

The Collobert & Weston model still provides no mechanism for handling word senses, but improves on the NPLM by adding several non-linear layers which increase its modelling capacity, and a convolutional layer for modelling longer range dependencies between words. Recursive neural nets (RNNs) directly address the problem of longer-range dependencies by allowing neighbour words to be combined into their phrasal equivalents in a bottom-up process.

The LBL model, despite its very simple linear structure, provides very good performance in terms of model

perplexity, but shares the problem of slow training and testing times and the inability to handle word senses or dependencies between words (outside its n-gram context).

In the HLBL model, Mnih and Hinton address the slow testing performance of the LBL model by using a hierarchical search tree over the vocabulary to exponentially speed up testing time, analogous to the concept of class-based language models (Brown et al., 1992). The HLBL model can also handle multiple word senses, but in their evaluation they show that in practice the model learns multiple senses (codes) for infrequently observed words instead of words with more than one meaning (Mnih and Hinton, 2009). The performance is strongly dependent on the initialisation of the tree, for which they present an iterative but non-optimal bootstrap-and-train procedure. Despite being non-optimal, it is shown to outperform the standard LBL model in terms of perplexity.

3 Mismatched Word Representations and Classifiers

The deep learning ideal is to train deep, non-linear models over large collections of unlabeled data, and then use these models to automatically extract information-rich, higher-level features[3] to integrate into standard NLP or image processing systems as added features to improve performance. However, several recent papers report surprising and seemingly contradicting results for this ideal.

In the most direct comparison for NLP, Turian (2010) compares features extracted using Brown clustering (a hierarchical clustering technique for clustering words based on their observed co-occurrence patterns), the hierarchical log-bilinear (HLBL) embeddings (Mnih and Hinton, 2007) and Collobert and Weston (C+W) embeddings (Collobert and Weston, 2008), by integrating these as additional features in standard supervised conditional random field (CRF) classification systems for NLP. Somewhat surprisingly, they find that using the more complex C+W and HLBL features do not improve significantly over Brown features. Indeed, under several conditions the Brown features give the best results.

These results are important for several reasons (we highlight these results in Table 2). The goal was to improve classification performance in structured prediction tasks in natural language by integrating features learned in a deep, unsupervised approach within a standard linear classification framework. Yet these complex, deep methods are outperformed by simpler unsupervised feature extraction methods.

[3] "Higher-level" features simply mean combining simpler features extracted from a text to produce conceptually more abstract indicators, e.g. combining word-indicators for "attack", "soldier", etc. to form an indicator for `WAR`, even though "war" is not mentioned anywhere in the text.

System	Dev	Test	MUC7
Baseline	90.03	84.39	67.48
HLBL 100-dim	92.00	88.13	75.25
C&W 50-dim	92.27	87.93	75.74
Brown, 1000 clusters	92.32	**88.52**	**78.84**
C&W 200-dim	**92.46**	87.96	75.51

Table 2: Final NER F1 results reported by Turian (2010).

In a sense, these seem to be negative results for the utility of deep learning in NLP. However, in this work we argue that these seemingly anomalous results stem from a mismatch between the feature learning function and the classifier that was used in the classification (and hence evaluation) process.

We consider the learning problem $h : \mathcal{X} \rightarrow \mathcal{Y}$ to decompose into $h = h'(\phi(\mathcal{X}))$, where ϕ is the feature learning function and h' is a standard supervised classifier. ϕ reads input from $\mathcal{X}$ and outputs encodings in feature space $\phi(x)$. h reads input in feature space $\phi(x)$ and outputs encodings in the output label space $\mathcal{Y}$.

Note that this easily extends to deep feature learning models by simply replacing $\phi(\mathcal{X})$ with $\phi^{(k)}(\cdots\phi^{(2)}(\phi^{(1)}(\mathcal{X}))\cdots)$, for a k-layer architecture, where the first layer reads input in $\mathcal{X}$ and each subsequent layer reads the output of the previous layer.

Within this view of the deep learning process, we can see that unsupervised feature learning does not happen in isolation. Instead, the learned features only make sense within some learning framework, since the output of the feature learning function ϕ (and each deep layer $\phi^{(k-1)}$) maps to a region in feature code space which becomes in turn the input to the output classifier h' (or subsequent layer $\phi^{(k)}$) . We therefore argue that in a semi-supervised or unsupervised classification problem, the feature learning function ϕ should be strongly dependent on the classifier h' that interprets those features, and vice versa.

This notion ties in with the standard deep-learning training protocol of unsupervised pre-training followed by *joint* supervised fine-tuning (Hinton et al., 2006) of the top classification layer and the deeper feature extraction layers. We conjecture that jointly training a deep feature extraction model with a linear output classifier leads to better linearly separable feature vectors $\phi(x)$ than training both independently. Note that this is in contrast to how Turian (2010) integrated the unsupervised features into existing NLP systems via disjoint training.

4 Proposed Work and Research Questions

For simpler sequence tagging tasks such as part-of-speech tagging and noun phrase chunking, the state-of-the-art models introduced in Section 2 perform adequately. However, in order to make use of the increased

modelling capacity of deep neural models, and to successfully model more complex semantic tasks such as anaphora resolution and semantic role labelling, *we hypothesise that the model needs to avoid modelling purely local lexical semantics and needs to efficiently handle multiple word senses and long-range dependencies between input words (or phrases) and output labels*. We propose to overcome the limitations of previous models with regard to these design goals, by focusing on the following key areas:

Input language representation: Neural models rely on vector representations of their input (as opposed to discrete representations as in, for instance, HMMs). In NLP, sentences are therefore encoded as real-valued embedding vectors. These vectors are learned in either a task-specific setting (as in the C+W model) or as part of a language model (as in the LBL model), where the goal is to predict the next word given the learned representations of the previous words. In order to maximise the information available to the model, we need to provide information-rich representations to the model. Current approaches represent each word in a sentence using a distinct word vector based on its literal string representation. However, as noted earlier, in NL the same words can have different senses based on the context in which it appears (polysemy). We propose to extend the hierarchical log-bilinear (HLBL) language model (see Section 2.2) in two important ways. We choose the HLBL model for its simplicity and good performance compared to more complex models.

Firstly, we propose to replace the iterative bootstrap-and-train process for learning the hierarchical tree structure over the vocabulary with a modified self-balancing binary tree. The tree rebalances itself from an initial random tree to leave most frequently accessed words near the root (for shorter codes and faster access times), while moving words between clusters to maximise overall model perplexity.

Secondly, we propose to add a word sense disambiguation layer capable of modelling long-range dependencies between input words. For this layer we will compare a modified RNN layer to a convolutional layer. The modified RNN will embed each focus word with its n most discriminative neighbour words (in a sentence context window) into a new K-dimensional, sense-disambiguated embedding vector for the focus word. We will evaluate and optimise the final model's learned representations by evaluating language model perplexity on held out data.

Model architecture and internal representation: Deep models derive their modelling power from their hierarchical structure. Each layer transforms the output representation of its previous layer, allowing the model to learn more general and abstract feature combinations in the higher layers which are relevant for the current task. The representations on the hidden layers serve as transformed feature representations of the input data for the output classifier. Enforcing sparsity on the hidden layers has been shown to produce stronger features for certain tasks in vision (Coates et al., 2010). Additionally, individual nodes might be highly correlated, which can also reduce the performance of certain classifiers which make strong independence assumptions (for instance naive Bayes). We propose to study the effect that enforcing sparsity in the learned feature representations has on task performance in NLP. Additionally, we propose to evaluate the effect that an even stronger training objective – one that encourages statistical independence between hidden nodes by learning factorial code representations (Hochreiter and Schmidhuber, 1999) – has on model performance.

Modelling structure in the output space: Tasks in NLP mostly involve predicting labels which exhibit highly regular structure. For instance, in part-of-speech tagging, two determiners have a very low likelihood of following directly on one another, e.g. "the the". In order to successfully model this phenomenon, a model must take into account previous (and potentially future) predictions when making the current prediction, e.g. as in hidden Markov models and conditional random fields. We propose to include sequential dependencies in the output labels and to compare this with including a convolutional layer below the output layer, for predicting output labels in complex NLP tasks such as coreference resolution and event structure detection.

5 Conclusion

Deep learning methods offer an attractive unsupervised approach for extracting higher-level features from large quantities of text data to be used for NLP tasks. However current attempts at integrating these features into existing NLP systems do not produce the desired performance improvements. We conjecture that this is due to a mismatch between the learned word representations and the classifiers used as a result of disjoint training schemes, and our thesis roadmap suggests three key areas for overcoming these limitations.

References

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

Y. Bengio. 2009. Learning deep architectures for ai. *Foundations and Trends® in Machine Learning*, 2(1):1–127.

P.F. Brown, P.V. Desouza, R.L. Mercer, V.J.D. Pietra, and J.C. Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

A. Coates, H. Lee, and A.Y. Ng. 2010. An analysis of single-layer networks in unsupervised feature learning. *Ann Arbor*, 1001:48109.

R. Collobert and J. Weston. 2008. 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.

R. Collobert. 2011. Deep learning for efficient discriminative parsing. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

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

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

G.E. Hinton, S. Osindero, and Y.W. Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

S. Hochreiter and J. Schmidhuber. 1999. Feature extraction through lococode. *Neural Computation*, 11(3):679–714.

A. Hyvärinen, J. Karhunen, and E. Oja. 2001. *Independent component analysis*, volume 26. Wiley Interscience.

I.T. Jolliffe. 2002. *Principal component analysis*, volume 2. Wiley Online Library.

J. Lafferty. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML, 2001*.

S. Lloyd. 1982. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137.

C.D. Manning, H. Schütze, and MITCogNet. 1999. *Foundations of statistical natural language processing*, volume 999. MIT Press.

A. Mnih and G. Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM.

A. Mnih and G.E. Hinton. 2009. A scalable hierarchical distributed language model. *Advances in neural information processing systems*, 21:1081–1088.

F. Morin and Y. Bengio. 2005. Hierarchical probabilistic neural network language model. In *AISTATS05*, pages 246–252.

F. Rosenblatt. 1957. *The Perceptron, a Perceiving and Recognizing Automaton Project Para*. Cornell Aeronautical Laboratory.

C.E. Shannon and W. Weaver. 1962. *The mathematical theory of communication*, volume 19. University of Illinois Press Urbana.

R. Socher, C.D. Manning, and A.Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

R. Socher, C.C. Lin, A.Y. Ng, and C.D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, volume 2, page 7.

J. Turian, L. Ratinov, and Y. Bengio. 2010. 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.