Natural languages understanding by a compositional alignment of word embeddings

by

Zining Zhu

Supervisor: Frank Rudzicz

April 10, 2019

Natural languages understanding by a compositional alignment of word embeddings

by

Zining Zhu

Supervisor: Frank Rudzicz

April 10, 2019

Abstract

Understanding the meaning of natural languages has been a fundamental problem in NLP research. A theory to understand language meanings widely accepted in linguistics is compositionality, stating that the meaning of any expression is determined by its structure and the meanings of its constituents. When an expression is analyzed in NLP, its structures can be captured by neural networks, while the meanings of its constituents can be represented by embedding vectors. Linguistic compositionality motivates our NLU framework.

Coming up with good neural network structures require tenuous hyper-parameter tuning, and figuring out novel embedding methods need significant insights. We would like to utilize existing embedding methods in a compositional framework to capture natural language meanings. In this work, we use consensus networks (Zhu et al., 2019) to find compositional alignments of existing word embeddings. Consensus networks project different views into a shared, indistinguishable, low-dimensional representation space, and regulate the procedure with an artificial noise view. In 7 out of 8 scenarios on two datasets, IMDB sentiment classification and SNLI textual entailment datasets respectively, our models outperform baseline models.

Acknowledgements

This thesis would not have been possible without the guidance and support of many people.

First, I want to give special thanks to my supervisor, Professor Frank Rudzicz, for supporting the works since I did not know much about NLP, discussing about research ideas, monitoring the qualities of research procedures, and proofreading my manuscripts even when he is super busy.

I am grateful to friends in SPOC lab that talked about relevant ideas. These discussions helped the formulation of many research ideas. A more comprehensive version of this thesis work will be submitted to a conference.

Part of formulation of this project should attribute to the APS281 course – Dr. Penny Kinnear, Ted, and Patrick led us to study and discuss about frameworks letting languages acquire meanings from applied linguistic viewpoints

This work is an extension of the consensus network project done in Winterlight Labs. Consensus network contains two papers: CN (Zhu et al., 2019) in NAACL and TCN (Zhu et al., 2018) in NeurIPS IRASL workshop. I want to say thank you to the Winterlight research director, Dr. Jekaterina Novikova, and science advisor, Prof. Rudzicz, for supervising this series of projects. Part of these works should be attributed to Winterlight colleagues – Liam, Maria, Jordan, Sasha, Aparna, Josh, Allen, Bill, Fariya, etc., who supported my work both during PEY year and afterwards.

The compositional embedding alignment project involved significant computational efforts. This could not have been possible without the cloud administrators, Dr. Relu Patrascu, who helped me set up and run experiments on both SPOC and Vector clusters.

I would like to thank all friends in Engineering Science. I enjoyed the four (plus one) years of undergraduate lives with you and will be proud to be an EngSci alumnus.

Finally I wish to express sincere gratitude to my family, for their love and support throughout the journey of learning and exploration.

Contents

List of Tables

List of Figures

Introduction

Understanding the meaning of natural languages has been a fundamental problem in natural languages processing research and linguistics. People have long realized that, in addition to describing experiences for people, languages are also able to shape the experiences for the speakers (Hoijer, 1954). This is referred to as linguistic relativity. Some people took a more radical step, hypothesizing that languages limit and determine cognitive categories. This is linguistic determinism. Linguistic relativity and determinism are treated as a weak and strong formulations of the Sapir-Whorf hypotheses, respectively (Pinker, 1994). Understanding the meanings of languages, therefore, is a significant step in knowing about the essence of intelligence, However, this has never been an easy task, because of the expressiveness and versatility of languages. "Their essence is hidden from us." (Wittgenstein, 1953).

Linguistic people have provided various frameworks to analyze mechanisms in which languages make sense to humans. Among them, linguistic compositionality is a prevalent framework, and can be related to NLP algorithms.

The linguistic compositionality framework is based on the compositionality principle. The principle has many formulations, and one of them can be briefly written as: In language L, the meaning of a complex expression ϵ is determined by its structure and the meanings of its constituents. This framework has various advantages. Following are two examples mentioned in Szabó (2017).

First, compositionality supports productivity. Given a set of syntactical rules and a finite vocabulary, we will be able to understand the meanings of sentences that we have not encountered before. This make possible for human to *learn* languages.

Second, compositionality breaks down meaning into syntactic and semantic aspects, which enables systemic analysis in natural languages processing. To capture the syntactic structures, people have assigned PoS tags and dependency parses to words (Aho

and Ullman, 1972). Neural networks (e.g., LSTM) are also used to capture syntactic structures (Kiperwasser and Goldberg, 2016; Kırnap et al., 2018). To capture the semantic of sentence components, people have used word-based features including tf-idf, distributed semantic models (Bengio et al., 2003; Mikolov et al., 2013; Bojanowski et al., 2017), and co-occurrence based models (Pennington et al., 2014; Shin et al., 2018). A challenging task is language modeling, which involves both syntactic and semantic aspect, but focuses on the discourse rather than understanding.

We follow the linguistic compositionality framework to build language understanding models. Although linguistic compositionality allows systematic NLP analysis, it is very hard to make developments in either syntactic or semantic aspect. Coming up with models that capture syntactic aspects by designing neural network structures require significant amount of experiments with trial-and-error. On the other hand, building new models with good grasp of semantic aspects requires great insights (Peters et al., 2018; Devlin et al., 2018; Howard and Ruder, 2018; Radford et al., 2019). Moreover, existing distributed semantic models suffer from least one of (1) explainability, (2) suboptimal distribution, and (3) lack of ability in handling non-standard tokens.

1.1 Thesis statement

The goal of this thesis is to develop a natural languages understanding (NLU) model using the linguistic compositionality framework. We utilize existing network structures to capture syntactic aspects, and use distributed semantic models (word embeddings) to capture semantics of individual words, and use a multi-view learning framework, consensus networks (Zhu et al., 2019), to combine the benefits of word embeddings.

We evaluate our compositional alignment models on two challenging NLU tasks, sentiment detection and inference classification. On 7 of 8 scenarios, the best compositional alignment models outperform the baseline.

1.2 Overview of the thesis

Chapter 2 will briefly review background works, including two linguistic frameworks and various NLP approaches organized according to compositionality framework cate-

gories: syntactic and semantic methods. Chapter 2 will also review multi-view learning methods that are relevant to the work in this thesis. These previous works motivate the development of our compositional alignment models. Chapter 3 proposes our models, and Chapter 4 describes the tasks we use to evaluate natural languages understanding (NLU) models. In Chapter 5, experiments and ablation studies are presented and discussed. Chapter 5 also discusses possible future works. This thesis concludes with a summary in Chapter 6.

Related works

2.1 Language meaning in linguistic frameworks

Two linguistic frameworks have been popular. Before mentioning the compositional framework mentioned above, we will first summarize another framework, meaning holism.

2.1.1 Holism

From the holism point of view, the meaning of individual words and sentences should never be understood in isolation. This framework could be illustrated by two arguments. First, the predecessor and successor sentences could qualify, even pivot the meaning of certain words in the present sentence. For example, the sentence "Meredith is in a terrible state" is ambiguous, dependent on the meaning of "state". If we appended a sentence "She would never want to travel to this state again", then the meaning of "state" as a mental status would be rendered inapplicable. Second, what the author wants to say depends on what is said, as well as what is not said. Discourses should be answered as "a stylistic whole". Especially, for any novel involving stories, the development of the world with time form a unique structure, *chronotope* (Bakhtin, 1935). As an example, the question "Why so serious?" immediately reminds people of Joker in the movie Dark Knight. In short, it is impossible to determine precise the meaning of words independent of their contexts.

Holism, however, is subject to various counter-arguments. Jackman (2017) mentioned several of them. On one hand, holism appears to deny the possibility of understanding languages, because the unspoken words could potentially lead to significant changes in language meanings. This is the argument of instability. On the other hand, holism seem to treat every sentence to have a positive impact on the holistic semantics,

whereas in fact some sentences are just objectively false. This is the argument of objectivity. More importantly, holism is in general incompatible with the compositionality assumption about languages: regardless of unspoken contexts, people are able to capture the meanings fairly well, if they parsed the sentences correctly and they understood individual word meanings.

2.1.2 Linguistic compositionality

According to Szabó (2017), the principle of compositionality has several slightly different formulations. One of them, the contextual variant, can be stated as: In language L, the meaning of a complex expression ϵ is determined by its structure and the meanings of its constituents under the *context* of ϵ.

Fodor and Lepore (1991) advocated the *reverse compositionality*: The meaning of a word is determined by any complex expressions in which the word occurs as a constituent. This argument, usually referred to as Frege's context principle, bridges the gap from "understanding sentences" to "understanding the words", which people in general take for granted.

2.2 Syntactic methods

To understand sentences syntactically, people have proposed various machine learning tasks including part-of-speech tagging, and dependency parsing. Note that many algorithms to solving these problems involve some semantic understanding of the sentences, but the following summary focus on their syntactic aspects.

2.2.1 Part-of-speech tagging

Part-of-speech (PoS) tagging assigns a part-of-speech tag to each word in the sentence. Usually the PoS tags follow the Penn TreeBank categories (Marcus et al., 1994), including `VB` (verb base form), `JJ` (adjective), and `NN` (singular form noun), etc. Another set of tags, the universal PoS tags[1], are also widely used. These include `NOUN` (noun), `ADJ` (adjective), and `PRON` (pronoun). SpaCy[2] produces universal PoS tags.

[1] http://universaldependencies.org/u/pos/
[2] https://spacy.io

PoS tagging is usually formulated as a sequential classification problem. Traditionally, hidden Markov models (HMMs) and Viterbi algorithm have been applied to address PoS taggings (Manning and Schütze, 1999). A popular solution towards sequential classification problem is to run an RNN across the sentence outputting softmax probabilities at each step. Nowadays, Bidirectional LSTM based models (Bohnet et al., 2018) have reached up to 97.96% accuracies on Penn TreeBank. However, PoS tagging is only a subtask in understanding natural languages.

2.2.2 Dependency parsing

Dependency parsing (DP) is another popular sub-task for NLU. In dependency parsing, people build up a syntactic tree for each sentence, where each word is a tree node. Note that there might be multiple possible parsing trees for syntactically ambiguous sentences. Probabilistic parsing therefore computes the conditional probability of each tree t for a sentence s given some grammar rules G: $P(t \mid s, G)$.

Dependency parsing could also be formulated as a sequential tagging problem. In spaCy, for example, each word is given two tags: `dep_` (the dependency tag, like `ROOT` or `nsubj`), and `head`, the parent node of the word in parsing tree. For each step, several possible choices could be assigned to the word in the sentence, as well as a conditional probability for the sentence segment $P(t_{1:i} \mid s_{1:i}, G)$, where $s_{1:i}$ represents the sentence segment and $t_{1:i}$ represents the set of tags that constitute the parse tree up till step i. In each step i, the choice of tag leads to potentially different possibilities of the following step: $P(t_{1:i+1} \mid s_{1:i_1}, G) = P(t_{1:i} \mid s_{1:i}, G)P(t_{i+1} \mid t_{1:i}, s_{1:i+1}, G)$. Iterating through the sentence eventually results in a joint probability $P(t \mid s, G)$.

Note that many of the conditional probability factors are shared, so dynamic programming could be applied to avoid many repeated computations. On the other hand, the sequential prediction problem gives a tree-structured search space. Various search methods (e.g., stack decoding (Jelinek, 1969), uniform-cost search, A*) could be applied to the search problem.

The current state-of-the-art dependency parsers are based on LSTMs (Clark et al., 2018). On Penn Treebank dataset, they reached 96.61 UAS (unlabeled attachment score) and 95.02 LAS (labeled attachment score).

2.3 Semantic methods

From the semantic aspect, most existing natural language understanding subtasks approximately belong to three categories: word features, language modeling, and embedding.

2.3.1 Words as features

The most simple approach to encode features is one-hot encoding. If we use a high-dimensional vector to represent a document d, each word in vocabulary corresponds to a dimension in vector. If we take the raw occurrences of each word ("term", $f_{t,d}$) as its corresponding feature value, frequent words that are not specifically associated to any documents like "the", "a" will have overwhelming large weights. An improved feature is the tf-idf, which is the product of term frequency and inverse document frequency. Term frequency is computed from the occurrences of words. A popular implementation for term frequency of term t in document d is:

$$TF(t,d) = 0.5 + \frac{f_{t,d}}{\max\{t' \in d : f_{t',d}\}}$$

where the additional 0.5 smoothed the distribution and prevents zero values in the out-of-document terms. Document frequency, on the other hand, can be implemented as:

$$IDF(t,D) = \frac{N}{1 + |\{d \in D : t \in d\}|}$$

where N is the total number of documents in the corpus D: $N = |D|$, and the term $|\{d \in D : t \in d\}|$ counts the number of documents where the term t occurs. The additional 1 prevents the denominator to be zero. Those words occurring in more documents will have lower inverse document frequency, hence their tf-idf feature values will be inhibited.

Usually this approach generate high-dimensional vectors (several thousands, depending on the size of vocabulary). SVM have been particularly well-suited for handling them (Joachims, 1999).

An important drawback for individual word-based features is their negligence of the effects of word co-occurrences on their semantics. In discourses, surrounding texts

produce an immediate context that could modify the meaning of individual words. For example, in "Meredith is in a terrible state, one I have never been *in*", the word "state" probably refers to some mental state. In "Meredith is in a terrible state, one I have never been *to*", the word "state" more likely refers to some geographical units. With $tf\text{-}idf$ representation, however, the feature of the word "state" would remain the same scalar. Given the same context, the vector representations of these two sentences might even be the same, because both "in" and "to" are *stopwords* have minimal inverse document frequencies. The discrepancies in these two sentences reside in the co-occurrences of words, which simple word-based features fail to capture.

2.4 Language modeling

Language modeling contains a wide variety of approaches. In general, they model the conditional probability of a word w within its context c_w: $P(w \mid c_w)$. Note that language modeling involves both syntactic and semantic aspects of languages.

If the context of a token (e.g., word) refers to all previous tokens in a sentence $w_{0..i-1}$, then the language model predicts the conditional likelihood of the token $P(w_i \mid w_{0..i-1})$. Char-RNN (Karpathy et al., 2015) and GPT (Radford et al., 2019) belong to this category. If otherwise, the context of a token refers to both previous and successor tokens in a sentence of length n tokens, then the language model predicts the cloze task conditional probability $P(w_i \mid w_{0..i-1}, w_{i+1,..n})$. BERT (Devlin et al., 2018) fall in this type.

2.4.1 N-gram

N-gram is a particular type of language models. N-gram models predict the probability of a word based on a window of $N-1$ previous words:

$$P(w_i \mid c_{w_i}) \approx P(w_i \mid w_{i-1}.., w_{i-N+1})$$

A benefit of N-gram models is their capture of word co-occurrence information. Therefore, frequencies of N-grams could represent their probabilities:

$$P(w_i, ..., w_{i+N-1}) \approx f(w_i, ..., w_{i+N-1})$$

Here, no *tf-idf* weighting is required. Note that N-gram could involve a much larger vocabulary space than individual words as features, and that most frequency entries would be zero due to the lack of coverage in text corpus. Smoothing methods like Kneser-Ney algorithm (Kneser and Ney, 1995) are usually applied when computing probabilities from frequencies.

2.4.2 Network-based LMs

Typical examples include OpenAI's Generative Pre-Training (GPT) (Radford et al., 2019), where a multi-layer model based on Transformer (Vaswani et al., 2017) with residual connections (He et al., 2016) learns the conditional probabilities and each token is a pair of bytes. Another outstanding example is the Bidirectional Encoders Representations from Transformers (BERT) model, which randomly masked 15% of words when passing into a deep bidirectional Transformer.

2.4.3 Embedding

A common problem in language modeling approaches mentioned above is the *curse of dimensionality*. The dimension of vectors is determined by the size of vocabulary, which could result in prohibitive computational efforts in various tasks.

Probabilistic neural representations as embedding Bengio et al. (2003) put forward a walkaround method. They used low-dimensional neural network representations for individual words, and optimize the network parameters towards maximizing the likelihood of occurring words. This is possible because neural networks are hierarchical feature extractors that can learn representative embedding spaces in their intermediate layers. In their paper, the loss function was a sum of the log probability and a regularization term $R(\theta)$:

$$\min_{\theta}\{-\log P_{\theta}(w_1, ..., w_n) + R(\theta)\}$$

where θ is the parameters in neural networks, and the regularization term is a weight decay penalty. When training with back propagation with parallel computation, the optimization could be completed within realistic time.

Word2vec embedding Word2vec was made popular by Mikolov et al. (2013). They use a two-layer neural network in either skip-gram (SG) or continuous-bag-of-words (CBOW) manner, and take the intermediate representation as embedding of each word. For SG, the network predicts the context of a word. For CBOW, the network predicts the current word based on its contexts. They used hierarchical softmax and negative sampling in implementation. An impressive property of word2vec models is their ability to map semantic similarity into Euclidean space, so that simple arithmetic addition / subtraction could correspond to the semantic analogy task, for example:

$$\text{MostSimilar}(v_{king} + v_{man} - v_{woman}) = v_{queen}$$

FastText (Bojanowski et al., 2017) incorporated some subword tokens, so many frequent typos would not result in out-of-vocabulary error wen querying word embeddings.

Word co-occurrence embedding Another category of methods to compute low-dimensional, semantic-preserving embedding vectors for words result from the co-occurrence statistics.

GloVe (Pennington et al., 2014) embeddings are based on the observations that similar words more likely to occur within the context of each other, and that the relative instead of absolute frequencies indicate the likelihoods. With some simplifications, GloVe proposed a global log-bilinear regression target, which leads to a set of low-dimensional word representations.

Other approaches utilize co-occurrence statistics. For exmaple, Shin et al. (2018) applied an SVD to the positive point-wise mutual information (PPMI) marix, and truncated a small number of columns as embeddings. An eigenvector analysis provided some hints towards the properties of the word distributions (e.g., a high eigenvalue of a word might indicate close relationship to corresponding semantic group), hence increasing the explainability.

Problems of embeddings Although word embedding methods have demonstrated impressive performances on various tasks, they are subject to numerous criticisms.

First, the latent features in embedding spaces are not explainable. This has been a problem with most latent, especially neural network derived features (Lipton, 2018).

Second, the distributions of embedded word vectors are far from optimal. For example, Gong et al. (2018) showed that many embeddings tend to place high frequency words and low frequency words in distant sub-regions. Caliskan et al. (2017) showed that word embeddings implicitly learn gender, race, and other bias that correspond to human results in the Implicit Association Test (Greenwald et al., 1998).

Third, word embeddings in general lack the ability to handle non-standard language tokens, including typos, short-hand notations, domain-specific vocabulary, and other language tokens that are either rare or absent from training corpus.

Each embedding could handle one or two of the above-mentioned drawbacks, but none of the existing word embeddings are simultaneously (1) inclusive of possible tokens, (2) agnostic of undesired information, (3) expressive enough for downstream tasks, (4) explainable, and (5) convenient for off-the-shelf usage. In this thesis, we use *multi-view learning* to mitigate the drawbacks of various word embeddings.

2.5 Multi-view learning

One formulation of multi-view learning can be usually formulated as: Given N data samples $\mathbf{x}^{(i)}|_{i=1..N}$, where each data point contains M views: $\mathbf{x}^{(i)} = [\mathbf{x_1}^{(i)}, .., \mathbf{x_M}^{(i)}]$. Each view could have different number of features, but all data points are constrained to these dimensions. The data labels are also given as $y^{(i)}$. We would like to learn a probability distribution $P(y|\mathbf{x})$.

2.5.1 Canonical correlation analysis

Traditionally, multi-view learning is addressed by canonical correlation analysis (CCA).

CCA The goal of CCA is to find a factor $\mathbf{z}$ that maximally correlates to two sets of variables $\mathbf{x_1}$ and $\mathbf{x_2}$. To use CCA in multi-view learning, two projection matrices w_1 and w_2 are set up (Hotelling, 1992):

$$w_1^*, w_2^* = \max_{w_1, w_2} \text{corr}\{w_1 x_1, w_2 x_2\}$$

The learned factors can be either of $w_1^* x_1$ and $w_2^* x_2$, where the w_i^* represent the optimal value for the weight parameter w_i. Note that although we described the case for two factors, the CCA method could be easily extended to multiple factors by for example maximizing the sum of correlations between all pairs of factors. Finally, the learned factors could be conveniently used for downstream classification tasks.

Kernel CCA Kernel canonical correlation analysis (KCCA) (Hardoon et al., 2004) projects the two input factors into latent spaces using functions f_1 and f_2. These spaces, written as $\mathcal{H}_1$ and $\mathcal{H}_2$, are the reproductive kernel Hilbert spaces (RKHS) of the functions f_1 and f_2 respectively. KCCA maximizes the correlations between the images of the two projection functions:

$$\begin{aligned} f_1^*, f_2^* &= \max_{f_1 \in \mathcal{H}_1, f_2 \in \mathcal{H}_2} \operatorname{corr}(f_1(\mathbf{x_1}), f_2(\mathbf{x_2})) \\ &= \max_{f_1 \in \mathcal{H}_1, f_2 \in \mathcal{H}_2} \frac{\operatorname{cov}(f_1(\mathbf{x_1}), f_2(\mathbf{x_2}))}{\sqrt{\operatorname{var}(f_1(\mathbf{x_1}) \operatorname{var}(f_2(\mathbf{x_2})))}} \end{aligned}$$

Note that the "kernel trick" is used here. Specifically, if $\phi(\mathbf{x_1})$ and $\phi(\mathbf{x_2})$ represent two sets of features in a metric space $\mathcal{H}$, then their inner product could be represented using a continuous valued kernel function $K(\mathbf{x_1}, \mathbf{x_2})$:

$$K(\mathbf{x_1}, \mathbf{x_2}) = \langle \phi(\mathbf{x_1}), \phi(\mathbf{x_2}) \rangle_{\mathcal{H}}$$

Deep CCA Using deep neural networks as encoders $E_{1..M}$ to project vectors $\mathbf{x}$ into latent representations $\mathbf{z}$, we can optimize the encoders to make the $\mathbf{z}$ maximally correlate to all views of features:

$$E_1, .., E_m = \max_{E_{1..M}} \operatorname{corr}\{\mathbf{z_{1..M}}\}$$

This is Deep CCA (Andrew et al., 2013). CCA-based approaches have been the default multi-view learning methods prior to generative adversarial networks (Goodfellow et al., 2014) became popular.

2.5.2 Invariant representation

In recent years, Generative Adversarial Network (GAN) (Goodfellow et al., 2014) motivates a category of multi-view learning approach: invariant representations. In GAN, a generator network tries to generate fake images that are indistinguishable from real images, while the discriminator network tries to tell real and fake images apart. Similarly, to produce invariant representations, several deep encoders try to project the features from different views onto an indistinguishable low-dimensional representational space, while a discriminator network tries to distinguish the originating view of the latent representations. The encoders $E_{1,..M}$ and the discriminator D are optimized with adversarial goals:

$$\max_{E_{1..M}} \min_{D} \mathcal{L}_D$$

where $\mathcal{L}_D$ can be the cross entropy loss of originating view classification. In practice, these goals can be realized with a gradient reversal layer, which can be easily implemented by various automatic differentiation toolboxes including TensorFlow (Abadi et al., 2016) and PyTorch (Paszke et al., 2017).

2.5.3 Relation to domain adaptation and transfer learning

Note that multi-view learning are closely related to domain adaptation and transfer learning, because of the similarity in tasks. Multi-view learning learns a shared representation from input data of different origin. Domain adaptation and transfer learning often require models to learn transferable representations of features between domains, which require the models to have capacity to extract information to a shared representational space. It is therefore no wonder that prevalent multi-view learning approaches are also widely used in domain adaptation and transfer learning (Long et al., 2015; Ganin et al., 2016; Hsu et al., 2018). In this thesis, we do not delve into the tasks of domain adaptation and transfer learning. We consider natural languages understanding a uni-domain problem, and do not transfer prior knowledge from pre-train corpus.

2.5.4 Consensus networks

A problem exists in learning adversarial view-invariant representations for multi-view learning. Although the latent representations are indistinguishable from the viewpoint of the discriminator network, there is no way we can control how the features are projected onto the representational spaces. This motivates our development of consensus networks (Zhu et al., 2018, 2019).

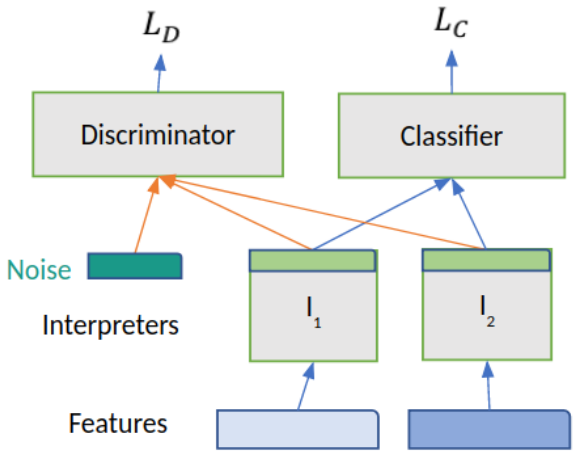

Figure 2.1: Consensus Network

In consensus networks, we divide linguistic features from different modalities (i.e., acoustic, syntactic, and semantic) $\mathbf{x_{1..M}}$ into non-overlapping views, and project them into indistinguishable, low-dimensional subspaces with a set of *interpreter* networks $I_{1..M}$. A discriminator D tries to tell apart their originating modalities (by minimizing $\mathcal{L}_D$), and a classifier C tries to classify based on the latent representations (by minizing $\mathcal{L}_C$). Both $\mathcal{L}_D$ and $\mathcal{L}_C$ are cross-entropy losses. Overall, the optimization goals are:

$$\min_{I_{1..M}} \max_{D} \mathcal{L}_D \qquad \text{and} \qquad \min_{C, I_{1..M}} \mathcal{L}_C \tag{2.1}$$

Three components in consensus networks are essential for their performance.

1. An artificial noise "view" is generated to pass in the discriminator but not the classifier network. This encourages the discriminator to carefully inspect the details of latent representations, and in turn encourages the interpreters to encode

rich representations. On DementiaBank, this improves classification accuracies by almost 5%. The most suitable type of noise depends on the problem scenarios. In Zhu et al. (2018), the mean and variance of the Gaussian noise is just the average of mean and variance of the true views. In our compositional alignment models, as will be shown in later chapters, a Gaussian noise with zero mean and a fixed variance turn out to be the most beneficial.

2. Cooperative optimization. We let gradients to back-propagate to the interpreter networks while optimizing $\mathcal{L}_C$. This guides the interpreters to encode information that are beneficial to multi-view learning.

3. Disentangled, iterative training procedure. Instead of using a gradient reversal layer, we let the two optimization mechanisms in 2.1, namely the consensus mechanism and classification mechanism, run iteratively. Therefore, only the classification needs training data labels, while the consensus mechanism can operate on abundant, less expensive, unlabeled training data. This results in Transductive Consensus Networks (Zhu et al., 2018).

2.6 Proposed methodology

The above related works naturally lead to the proposal of our framework. We propose an NLU framework derived from the linguistic compositionality framework. We use LSTMs (Hochreiter and Schmidhuber, 1997) to capture syntactical structures, use existing word embeddings to capture semantic of their components, words, and use consensus networks to gather the benefits of off-the-shelf the embeddings.

Language understanding models

There are two options when determining the order between the alignment with consensus networks and the recurrence with LSTM. We refer to them as AlignEmbed and AlignAttention, respectively. If we first align the embeddings and then pass into LSTM, the resulting model is AlignEmbed. If we first pass individual embeddings into LSTM, and then align with consensus networks, the model is AlignAttention. Both models, as well as the baseline (attentional LSTM classifier), use `AttentionalLSTM` as a building block. In this section, we briefly describe all three models.

3.1 Attentional LSTM Classifier

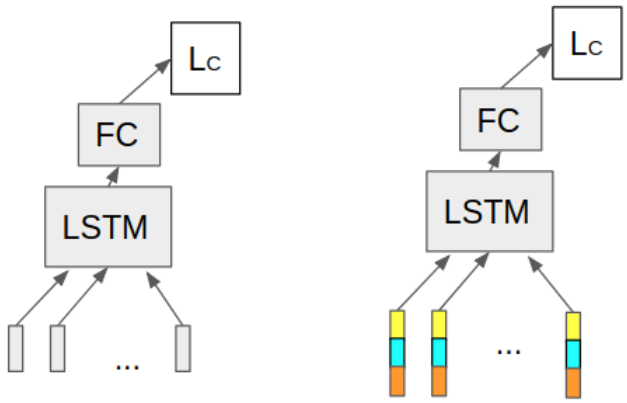

Figure 3.1: Basic LSTM Classifier model with one embedding (left) and three embeddings concatenated (right)

We use attentional LSTM (similar to (Yang et al., 2016)) to build a baseline classifier. Given a document $w_{1..T}$, we first acquire their vector representations $v_i^{(1)}..v_i^{(E)}$ through

E embeddings in off-the-shelf manner. For the i^{th} word, directly concatenating all embedding vectors results in the input vector $\mathbf{v_i}$. For a document containing N tokens (words), the queried vectors form a matrix with shape $(N, 300 \times E)$ (assuming each word embedding dimension 300). This the input matrix for the attentional LSTM.

The attentional LSTM model contains a long-short term memory (LSTM) network with attention mechanisms. The LSTM equations are summarized in Equation 3.1.

$$\begin{aligned}
\mathbf{i_t} &= \sigma(W_i[\mathbf{x_t}, \mathbf{h_{t-1}}] + b_i) \\
\mathbf{o_t} &= \sigma(W_o[\mathbf{x_t}, \mathbf{h_{t-1}}] + b_o) \\
\mathbf{f_t} &= \sigma(W_f[\mathbf{x_t}, \mathbf{h_{t-1}}] + b_f) \\
\mathbf{g_t} &= \tanh(W_g[\mathbf{x_t}, \mathbf{h_{t-1}}] + b_g) \\
\mathbf{c_t} &= \mathbf{f_t} \odot \mathbf{c_{t-1}} + \mathbf{i_t} \odot \mathbf{g_t} \\
\mathbf{h_t} &= \mathbf{o_t} \odot \tanh(\mathbf{c_t})
\end{aligned} \tag{3.1}$$

Here σ is the Sigmoid function. W_i, W_o, W_f, W_g are trainable parameter matrices of shape (d_{in}, d_{rnn}), mapping the input dimension d_{in} to hidden dimension d_{rnn}. Note that we use the bidirectional version, and the state outputs from both directions are concatenated to form a new state output: $\mathbf{h_t} = [\overrightarrow{\mathbf{h_t}}, \overleftarrow{\mathbf{h_t}}]$. In case multiple LSTM layers are placed, the state output $\mathbf{h_t}$ of lower layer is passed as the state input $\mathbf{x_t}$ into the upper layer LSTM. The state outputs of the final level LSTM are used to compute the attentions.

In this thesis, we use a simple attention mechanism, formulated in Equation 3.2.

$$\begin{aligned}
\mathbf{w} &= \mathbf{H}W \\
\mathbf{A} &= \mathbf{w}^T\mathbf{H}
\end{aligned} \tag{3.2}$$

where $\mathbf{H}$ is a matrix of shape (N, d_h), where d_h is the dimension of $\mathbf{h_t}$. Each row of $\mathbf{H}$ consists of the vector $\mathbf{h_t}$. The self-attention weight $\mathbf{w}$ is of shape (N, d_w), where d_w is a model hyperparameter. When multiplied with the matrix $\mathbf{H}$, the resulting attention $\mathbf{A}$ has shape (d_w, d_h).

Above mentioned is the structure of an `AttentionalLSTM` block, which will also be used in our following models.

The final step is to place a dense projection layer on top of attentional LSTM block.

This layer, referred to as `FC` in Figure 3.1 and the C network in following equations, predicts the probability of a document label from both the attentions and the final steps of LSTM:

$$P = \sigma(C[\mathbf{A}, \mathbf{h_N}, \mathbf{c_N}])$$

In training, we optimize all model parameters to minimize the cross-entropy loss:

$$\min \mathcal{L}_c, \text{ where } \mathcal{L}_c = \mathbb{E}\ \text{-log}P(\hat{y} = y)$$

3.2 AlignEmbed

This section describes the first of our two models, AlignEmbed.

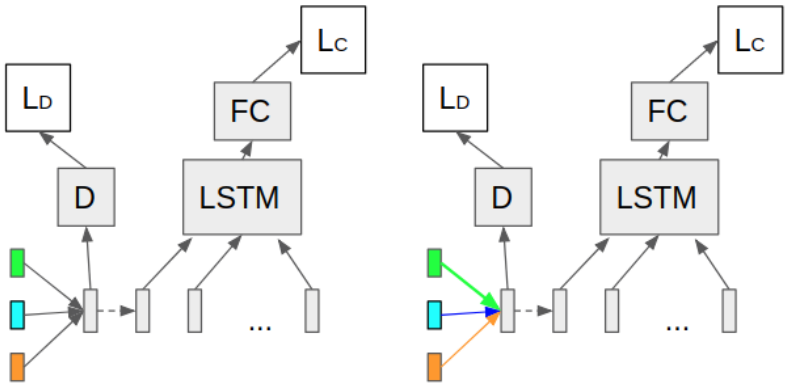

Figure 3.2: AlignEmbed model. Left: shared encoder parameters; right: independent encoders.

We use consensus networks (Zhu et al., 2019) to project vectors of word from multiple embedding domains, $\mathbf{v_i^{(e)}}$ into a shared low-dimensional space $\mathbf{x_i}$ before passing into the `AttentionalLSTM` classifier, in following three steps.

1. For a word vector $\mathbf{v_i^{(e)}}$ from different embeddings ($e = 1, 2, ...N_e$), an encoder E_e projects it to $\mathbf{x_i^{(e)}}$:

   $$\mathbf{x_i^{(e)}} = E_e(\mathbf{v_i^{(e)}})$$

   Dependent on whether these encoders share parameters, AlignEmbed has two versions, as denoted in Figure 3.2. Then concatenate all these N_e representations:

$\mathbf{x_i} = [\mathbf{x_1}, .., \mathbf{x_{N_e}}]$.

2. A discriminator network D is set up to classify which originating embedding does a word come from.

$$P_{disc}(\hat{e}) = \sigma(D(\mathbf{x_i^{(e)}}))$$

The corresponding loss is $\mathcal{L}_\mathcal{D} = \mathbb{E}_\mathbf{v}$-log $P_{disc}(\hat{e} = e)$.

3. According to the set up in consensus networks, an artificially generated noise embedding is also passed into the discriminator (but not the `AttentionalLSTM`), to regularize how the embeddings are projected. For example, if the generated noise embedding is Gaussian with large variance, then the projected embeddings are encouraged to be compact in variance.

An `AttentionalLSTM` module takes in the projected (and concatenated) embeddings, and outputs the attention for a fully connected classifier C to compute the probabilities. The equations defining the output and loss are abbreviated as:

$$\begin{aligned}\mathbf{a} &= \text{AttentionalLSTM}(\mathbf{x_{1..N}}) \\ P &= \sigma(C(\mathbf{a})) \\ \mathcal{L}_c &= \mathbb{E}_\mathbf{v} \text{ -log}P(\hat{y} = y)\end{aligned}$$

We set up competing objectives to optimize these two losses:

$$\min_E \max_D \mathcal{L}_D \text{ and } \min_{C,E} \mathcal{L}_C$$

3.3 AlignAttention

AlignAttention first compute a fixed-length attention vector $\mathbf{a^{(e)}}$ using the `AttentionalLSTM` block for each word embedding (indexed by e):

$$\mathbf{a^{(e)}} = \text{AttentionalLSTM}(\mathbf{v_i})$$

Then align the attentions using consensus network structure in the following three steps.

1. Let a discriminator network D guess the originating embedding index $\hat{e}$ of each attention. The equation and associated discrimination loss can be written as:

$$P(\hat{e}_e) = D(\mathbf{a}^{(e)})$$

$$\mathcal{L}_D = \mathbb{E}\ \text{-log}P(\hat{e} = e)$$

   Dependent on whether we share the attentional LSTM, there are two variants, as shown in Figure 3.3.

2. For each document with label y, all projected attentions are concatenated and passed into the fully-connected classifier C, which predicts the estimated output $\hat{y}$. The equation and associated classification loss are:

$$P(\hat{y}) = \sigma(C([\mathbf{a_1}, \mathbf{a_2}, ..]))$$

$$\mathcal{L}_C = \mathbb{E}\ \text{-log}P(\hat{y} = y)$$

3. An artificial "noise attention" is also passed into the discriminator (but not the classifier), to regularize how the attentions are projected to the common representational space.

Similar to AlignEmbed, the overall optimization goals is also a min-max game and a classification mechanism.

$$\min_D \max_\theta \mathcal{L}_D \qquad \text{and} \qquad \min_{\theta,C} \mathcal{L}_C$$

where θ represent the parameters of the `AttentionalLSTM` blocks.

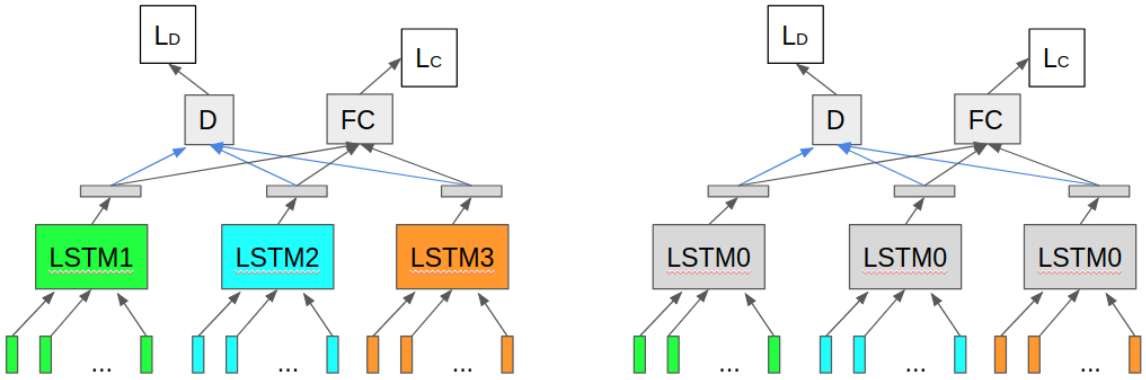

Figure 3.3: AlignAttention model. The left variant uses one attentional LSTM per embedding, while the right variant uses a shared attentional LSTM for different embeddings.

Evaluating the understanding

In this thesis, we evaluate the ability of natural languages understanding (NLU) with two widely used tasks: sentiment classification and textual entailment. Note that each of these tasks only measure an aspect of natural languages understanding abilities, which we will briefly describe in this section.

4.1 Sentiment classification

A popular sentiment classification dataset is IMDB movie review dataset (Maas et al., 2011). It contains 50,000 movie reviews from IMDB, with exactly half of them labeled as "positive" and half "negative". Following are two examples:

Positive (excerpt from train/pos/14_10.txt) `This film has a special place in my heart ... <br /><br /> Excellent performances by Jane Fonda and Robert DeNiro that rank with their best work, a great turn by a young Martha Plimpton, an inspiring story line, and a haunting musical score makes for a most enjoyable and rewarding experience.`

Negative (excerpt from `train/neg/13_2.txt`) `Weak plot, predictable violence, only semi interesting characters. Like the writer (also one of the stars?) was fictionalizing his own screw ups and added an incredulous fantasies of drugs and murder to make it "hot". From the predictable rap and house soundtrack, to the family conflicts, it's poorly acted, stereotypical, and ultimately terribly boring. ...`

Figure 4.1 contains a histogram of movie review document lengths (number of words in raw documents). 49,977 (99.95%) out of 50,000 movie reviews have lengths shorter

than 1200 tokens.

The current state-of-the-art performance is 95.3% accuracy, achieved by Howard and Ruder (2018).

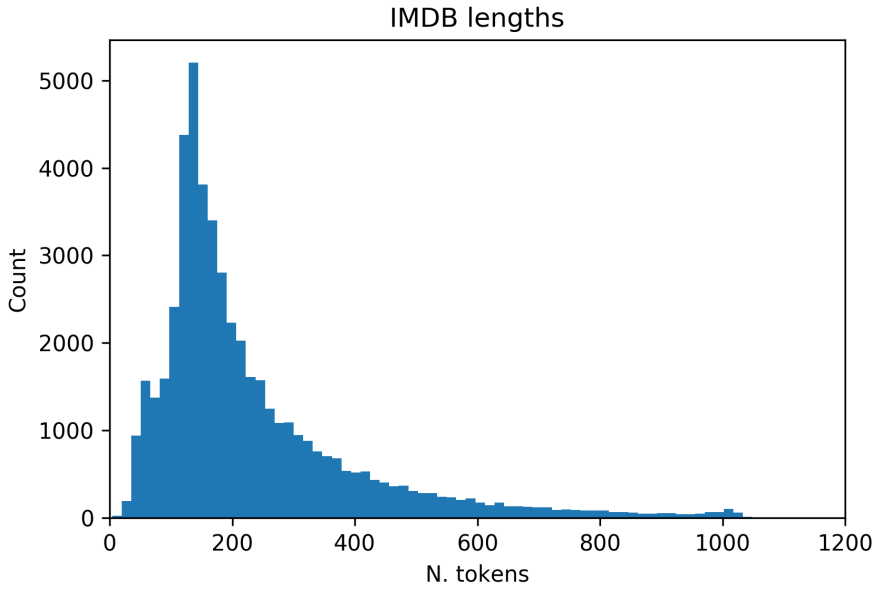

Figure 4.1: Histogram of review lengths for IMDB.

4.2 Textual entailment

A widely used textual entailment dataset is Stanford Natural Languages Inference (SNLI) dataset (Bowman et al., 2015b). The dataset contains 570,152 pairs of sentences, with label depicting their relations: "neutral" (189,218), "contradiction" (190,113), and "entailment" (189.702). For each data point, they acquired five human labels and took the majority. Note that there are 1,119 sentence pairs with ambiguous labels. These labeles are determined "ambiguous" because the five human labels fail to give a majority agreement. In our experiments, we exclude these sentence pairs.

Table 4.1 contains several SNLI examples. Note that the sentences might not be syntactically correct or complete, but the sentences carry certain amount of meaning. Figure 4.2 shows histograms of lengths (number of words) in two sentences of SNLI.

99.76% of sentences have lengths smaller than 50, and 98.66% have lengths smaller than 40.

Baseline LSTM encoders give around 80% accuracies (Bowman et al., 2015a, 2016). LSTMs with carefully designed self-attentions or hierarchical structures give 83% to 87% accuracies (Liu et al., 2016; Chen et al., 2018; Talman et al., 2018). Currently, the best results for SNLI are achieved by multi-task learning (Liu et al., 2019).

Sentence 1	Sentence 2	Label
Children smiling and waving at the camera.	There are children present.	entailment
Children smiling and waving at the camera.	They are smiling at their parents.	neutral
Children smiling and waving at the camera.	The kids are frowning.	contradiction
A couple walks hand in hand down a street.	A couple is walking together	entailment
A couple walks hand in hand down a street.	The couple is married.	neutral
A couple walks hand in hand down a street.	The couple is sitting on a bench.	contradiction
During calf roping a cowboy calls off his horse.	A first time roper falls off his horse.	neutral
During calf roping a cowboy calls off his horse.	Cowboy falling off horse.	(ambiguous)
During calf roping a cowboy calls off his horse.	A man ropes a calf successfully.	contradiction
An older women tending to a garden.	The lady is cooking dinner.	contradiction
An older women tending to a garden.	The lady is weeding her garden.	neutral
An older women tending to a garden.	The lady has a garden.	entailment

Table 4.1: SNLI examples

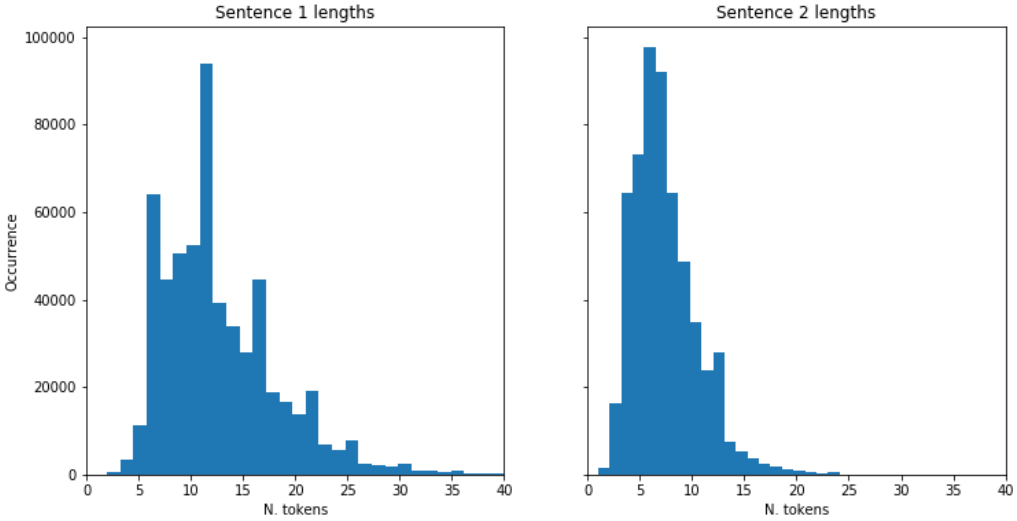

Figure 4.2: Histograms of SNLI sentence lengths.

Experiments and discussions

This section presents our experiment setting and results for the two tasks mentioned above: word similarity, sentiment classification, and textual entailment.

5.1 Movie review sentiment analysis

5.1.1 Setting

The IMDB dataset (Maas et al., 2011) is divided into training and testing sets, each with 25,000 samples. For evaluation, we randomly split the testing folder into `dev` and `test` set, respectively. Preprocessing includes removing the new-line html tag (`<br />`), punctuations, symbols, and the components with part-of-speech tagged as "unknown" by SpaCy [1]. All words are converted to lower-case.

Our models require multiple pretrained word embeddings. We use three embeddings: FastText[2] (Bojanowski et al., 2017), GloVe[3] (Pennington et al., 2014), and Word2vec[4] (Mikolov et al., 2013). All three word embeddings are downloaded from websites in an off-the-shelf manner. In the remaining sections of this thesis, they are referred to as "F", "G", and "W", respectively.

5.1.2 Results

The accuracy results are shown in Table 5.1. AlignEmbed models outperform remaining ones in all four combinations of embeddings.

[1] spacy.io

[2] FastText use `wiki-news-300d-1M-subword` from `fasttext.cc`.

[3] GloVe use `glove.840B.300d` from `https://nlp.stanford.edu/projects/glove/`.

[4] Word2Vec use `Model 6` (gensim-skipgram-wiki2017) from `http://vectors.nlpl.eu/repository`

Model	Embedding			
	F+W	F+G	W+G	W+F+G
LSTMClassify	.892	.914	.909	.915
AlignEmbed (shared projector)	**.910**	.916	.916	.914
AlignEmbed (separate projector)	.500	**.918**	**.917**	**.916**
AlignAttention (shared LSTM)	.905	.910	.900	.868
AlignAttention (separate LSTM)	.903	.904	.906	.874

Table 5.1: Classification accuracies on IMDB dataset.

5.1.3 Effect of lemmatization

We experiment on the effect of preprocessing inputs as an ablation study. Here, in additional to the previously mentioned preprocessing steps (i.e., removing punctuations, converting to lower cases), all movie reviews are lemmatized. Table 5.2 shows that accuracies drastically decrease in any choice of word embedding scenarios. Note that all three embeddings we use were trained on *unlemmatized* corpus. Therefore, a decrease in accuracy is expected, because of the mismatch in word forms (i.e., lemmatization or not) between the embedding corpus and train / test corpus.

Model	Embedding			
	F+W	F+G	W+G	W+F+G
LSTMClassify	.770	.773	.791	.768
AlignEmbed (shared projector)	.764	.790	.768	.797
AlignEmbed (separate projector)	.784	.633	.791	.800
AlignAttention (shared LSTM)	.798	.792	.807	.800
AlignAttention (separate LSTM)	.779	.801	.792	.789

Table 5.2: Classification accuracies on IMDB dataset (lemmatized).

5.2 Textual entailment classification

5.2.1 Setting

For each sentence pair in Stanford natural languages inference (SNLI) dataset, we concatenate the two sentences while inserting a separator punctuation token (e.g., the `period` sign at the end of sentence) in between. Additionally, we fix the length of each

input sequence to 40 by either padding with dummy words or truncating. In implementation, the sequence of concatenation was arbitrarily determined to be "sentence 2 - period - sentence 1", so that the padding or truncation will mainly affect "sentence 1", which is usually the longer one, as illustrated in Figure 4.2.

5.2.2 Results

As shown in Table 5.3, AlignEmbed models outperform AlignAttention and baseline LSTM classifier in most scenarios.

Model	Embedding			
	F+W	F+G	W+G	W+F+G
LSTMClassify	**.778**	.731	.774	.782
AlignEmbed (shared projector)	.773	.779	.743	**.786**
AlignEmbed (separate projector)	.774	**.780**	**.777**	.784
AlignAttention (shared LSTM)	.752	.750	.769	.741
AlignAttention (separate LSTM)	.770	.762	.771	.758

Table 5.3: Classification accuracies on SNLI dataset.

5.3 Discussions

5.3.1 Compositional alignments outperform baseline

From the above results, the better of our two compositional NLU models, AlignEmbed, outperforms benchmark LSTM in 7 out of 8 scenarios. Among the AlignEmbed models, the variant with separate projectors outperform baseline classifiers in 6 out of 8 scenarios. These experiments show that aligning embeddings with our compositional method could be superior to the alignments relying on deep LSTMs.

5.3.2 More embeddings do not guarantee better performances

In either of IMDB or SNLI experiments, the performances of models do not always improve as more embeddings are included. For example, in IMDB experiments (Table 5.1), W+F+G embeddings fail to outperform F+G in 4 of the 5 models. A possible

explanation to this effect is that proper projections of embeddings onto shared latent spaces could be hard, if the embeddings encode distributed semantics in very distinct manners.

Note that baseline LSTMs also do not guarantee better performances with additional embeddings. For example, on non-lemmatized experiments shown in Table 5.2, the accuracy of W+F+G fail to outperform either of F+W, F+G, or W+G.

5.3.3 Artificial Gaussian noise is recommended

The artificial noise view in consensus networks regularize how the views are projected onto the shared latent spaces. On the IMDB and SNLI experiments, we find a Gaussian noise with zero mean and variance 2.0 performing the best in most situations. Gaussian noise with variances 0.1 and 0.01 could sometimes lead to the model being trapped in local minimums, where the n-class training cross entropy loss equals $log(n)$ and accuracy on balanced n-class dataset equals $\frac{1}{n}$.

As a side note, noise views drawn from Gumbel and Rayleigh distributions (centered at 0, and scales range from 0.01 to 1.0, for both distributions) often lead to these local minimum scenarios.

5.3.4 Variables are controlled in experiments

The purpose of our work is not to design network structures that are specifically suitable for some tasks. Instead, we show the efficacy of our NLU framework by controlling a series of hyper-parameters throughout our experiments.

Structural prior For all models we implement (LSTMClassify, AlignEmbed and AlignAttention), we use the same attentional LSTM block to capture syntactic structures within the expressions. In all models relating to consensus networks (CNs), we let most CN-specific hyper-parameters be their default values.[5] In other words, we do no impose structural priors (other than what is inherent in our proposed framework) when building the models.

[5]These CN-specific hyper-parameters include the structure of discriminator (i.e., 1-layer fully connected network), and the manner of iterative training between the consensus and the classification mechanisms (i.e., set up consensus and classification mechanisms by 1:1). Zhu et al. (2018) discussed some alternative optimization methods.

Knowledge prior Different from many state-of-the-art natural language understanding models, we do not explicitly incorporate prior knowledge from large, pre-training corpus like BERT (Devlin et al., 2018) and GPT (Radford et al., 2019). Instead, we use existing word embeddings in an off-the-shelf manner. Neither do we fine-tune (i.e., allow gradients to backprop to) the embeddings like ULMFit (Howard and Ruder, 2018).

By controlling the knowledge prior as such, we compromise some accuracy in classification, but achieve something more important: cleaner evaluation of model capacities. Transferred knowledge is hard to control. Sometimes the models preserve too much knowledge from previous, heterogeneous tasks, resulting in negative transfer, while sometimes the models forget too much information, giving catastrophic forgetting (McCloskey and Cohen, 1989). By controlling the knowledge prior, we are able to analyze the effectiveness of our NLU models against baselines without being impacted by the relatively hard-to-control transferred knowledge.

Model specific hyper-parameters We tune the model-specific hyper-parameters of all models in the same, systematic manner, and report the best performance in each problem scenario. The optimal parameters for all scenarios in our experiments are included in Appendix A.

Training methods We follow the same training routine for all models. In all types of optimizations, we use Adam optimizer (Kingma and Ba, 2014) with a learning rate 10^{-5} and zero weight decay. Other Adam optimizer parameters follow PyTorch default (Paszke et al., 2017). The learning rate and weight decay are decided based on several priliminary experiments with LSTMClassify and AlignAttention on IMDB dataset. Each optimization is run for 100 steps, and the result from the step with the lowest *validation* loss is recorded.

5.4 Future Work

While our researches establish a compositional alignment NLU framework, Future works could be taken in several avenues.

First, we could extend the analysis and evaluate our compositional alignment NLU framework more thoroughly using both downstream tasks and intrinsic evaluations.

Potential downstream tasks include semantic role labeling (He et al., 2017). Possible intrinsic evaluation methods include SimLex-999 (Hill et al., 2015) and SimVerb-3500 (Gerz et al., 2016), which measure the model's abilities to map semantic similarity of individual words into numerical space. These analysis could further and more comprehensively evaluate the natural languages understanding capacities of our models.

Second, more intuitions could be drawn from linguistics and philosophy. Linguistic compositionality is not the only framework to analyze language and meanings. Semantic holism, for example, considers the whole language context to have impact on meanings of individual expressions. Pragmatics, as another example, considers the meanings of languages when they are used in social interactions. Both of them could be considered to motivate the developments of new natural languages understanding frameworks.

Conclusion

In this thesis, we propose a natural languages understanding (NLU) framework inspired by linguistic compositionality, which states that language meanings depend on sentence structures and the meanings of component words. We use bidirectional LSTMs to capture syntactic aspects, and use off-the-shelf word embeddings to gather word meanings. We use consensus networks (Zhu et al., 2019) to align various existing embedding methods in two methods: AlignEmbed and AlignAttention. On two NLU evaluation tasks, sentiment detection and textual entailment classification, our best compositional alignment models outperform baseline attentional LSTM classifier with the same word embeddings but concatenated. Our compositional alignment framework could serve as an ensemble method to utilize the benefits of multiple existing but imperfect word embeddings. In the future, more NLP algorithms inspired by linguistic and philosophical frameworks could be applied to build NLU models.

Bibliography

Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283.

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis. *Proceedings of the 30th International Conference on Machine Learning*, 28(3):1247–1255.

Mikhail Bakhtin. 1935. Discourse in the Novel. *Literary theory: An anthology*, 2:674–685.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

Bernd Bohnet, Ryan McDonald, Gonçalo Simões, Daniel Andor, Emily Pitler, and Joshua Maynez. 2018. Morphosyntactic tagging with a meta-bilstm model over context sensitive token encodings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2642–2652. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015a. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Samuel R Bowman, Gabor Angeli, Potts Christopher, and Christopher D Manning. 2015b. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. *arXiv preprint arXiv:1603.06021*.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Qian Chen, Zhen-Hua Ling, and Xiaodan Zhu. 2018. Enhancing sentence embedding with generalized pooling. *arXiv preprint arXiv:1806.09828*.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, Brussels, Belgium. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Jerry Fodor and Ernest Lepore. 1991. Why meaning (probably) isn't conceptual role. *Mind and language*, 6(4):328–43.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Victor Lempitsky, Urun Dogan, Marius Kloft, Francesco Orabona, and Tatiana Tommasi. 2016. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17:1–35.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simverb-3500: A large-scale evaluation set of verb similarity. *arXiv preprint arXiv:1608.00869*.

ChengYue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2018. Frage: Frequency-agnostic word representation. In *NeurIPS*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Proceedings of Advances in neural information processing systems*, pages 2672–2680.

Anthony G Greenwald, Debbie E McGhee, and Jordan LK Schwartz. 1998. Measuring individual differences in implicit cognition: the implicit association test. *Journal of personality and social psychology*, 74(6):1464.

David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. 2004. Canonical Correlation Analysis: An Overview with Application to Learning Methods. *Neural Computation*, 16(12):2639–2664.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada. Association for Computational Linguistics.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Harry Hoijer. 1954. The Sapir-Whorf Hypothesis. *Language in culture*, pages 92–105.

Harold Hotelling. 1992. Relations between two sets of variates. In *Breakthroughs in statistics*, pages 162–190. Springer.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 328–339.

Yi-Te Hsu, Zining Zhu, Chi-Te Wang, Shih-Hau Fang, Frank Rudzicz, and Yu Tsao. 2018. Robustness against the channel effect in pathological voice detection. *NeurIPS ML4H Workshop*.

Henry Jackman. 2017. Meaning Holism. In Edward N Zalta, editor, *The Stanford Encyclopedia of Philosophy*, spring 2017 edition. Metaphysics Research Lab, Stanford University.

F. Jelinek. 1969. Fast sequential decoding algorithm using a stack. *IBM J. Res. Dev.*, 13(6):675–685.

Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference of Machine Learning (ICML-99)*, volume 99, pages 200–209.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *ICLR*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Proceedings of the 1st International Conference on Learning Representations (ICLR)*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Ömer Kırnap, Erenay Dayanık, and Deniz Yuret. 2018. Tree-stack LSTM in transition based dependency parsing. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 124–132, Brussels, Belgium. Association for Computational Linguistics.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *ICASSP*.

Zachary Chase Lipton. 2018. The mythos of model interpretability. *Commun. ACM*, 61:36–43.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional lstm model and inner-attention. *arXiv preprint arXiv:1605.09090*.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. 2015. Learning Transferable Features with Deep Adaptation Networks. *arXiv*, 37.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 114–119, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Advances in neural information processing systems*, pages 1–9.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Steven Pinker. 1994. *The language instinct.* William Morrow & Co, New York, NY, US.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2019. Improving Language Understanding by Generative Pre-Training. Technical report, Open AI.

Jamin Shin, Andrea Madotto, and Pascale Fung. 2018. Interpreting Word Embeddings with Eigenvector Analysis. *NeurIPS IRASL Workshop*.

Zoltán Gendler Szabó. 2017. Compositionality. In Edward N Zalta, editor, *The Stanford Encyclopedia of Philosophy*, summer 2017 edition. Metaphysics Research Lab, Stanford University.

Aarne Talman, Anssi Yli-Jyrä, and Jörg Tiedemann. 2018. Natural language inference with hierarchical bilstm max pooling architecture. *arXiv preprint arXiv:1808.08762*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *NIPS*.

Ludwig Wittgenstein. 1953. *Philosophical Investigations*. Wiley-Blackwell.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Zining Zhu, Jekaterina Novikova, and Frank Rudzicz. 2018. Semi-supervised classification by reaching consensus among modalities. *NeurIPS IRASL Workshop*.

Zining Zhu, Jekaterina Novikova, and Frank Rudzicz. 2019. Detecting cognitive impairments by agreeing on interpretations on linguistic features. *NAACL*.

Appendix

A.1 Model hyper-parameters for IMDB experiments

Table A.1 lists the hyper-parameters for our best models in IMDB sentiment analysis experiments.

Note that there is a *seqlen* and a *step* hyper-parameter. These come from the way we divide the movie reviews into fixed-length sequences for recurrent neural networks. For each document, we take sequence of length *seqlen* as an input data sample, and slide backwards by step size *step* to take the next sequence. This step is repeated until there is not enough words for the next sequence in the document. All of the sequences are given the label of the document. In case the whole document contains less than *seqlen* words, we pad dummy words (e.g., "pad") to the end of the document to make its length equal to *seqlen*. During inference, all sequences from one document go through a "majority voting" process to determine the predicted probability of the document. To prevent training on testing data, we make sure that none of the sequences from the same document spread across train / dev / test sets.

A.2 Model hyper-parameters for SNLI experiments

Table A.2 lists the hyper-parameters for our best models in SNLI textual entailment analysis experiments. Note that each sentence pair is either padded or truncated to one sequence of a fixed length (*seqlen*). There is no majority voting here.

Model	layers	d_{lsrm}	*seqlen*	*step*	noise
LSTMClassify (F+G)	1	128	300	100	N/A
LSTMClassify (F+W)	1	128	300	100	N/A
LSTMClassify (W+G)	1	128	300	100	N/A
LSTMClassify (W+F+G)	1	128	300	100	N/A
AlignEmbed-shared (F+G)	1	128	300	100	No noise
AlignEmbed-shared (F+W)	1	128	300	100	Gaussian (var=0.5)
AlignEmbed-shared (W+G)	1	128	100	50	Gaussian (var=2.0)
AlignEmbed-shared (W+F+G)	1	128	300	100	No noise
AlignEmbed-sep (F+G)	1	128	300	50	Gaussian (var=2.0)
AlignEmbed-sep (F+W)	1	128	300	100	Gaussian (var=2.0)
AlignEmbed-sep (W+G)	1	128	100	50	Gaussian (var=2.0)
AlignEmbed-sep (W+F+G)	1	128	300	100	Gaussian (var=2.0)
AlignAttention-shared (F+G)	1	128	300	100	No noise
AlignAttention-shared (F+W)	1	128	300	100	Gaussian (var=2.0)
AlignAttention-shared (W+G)	1	128	300	100	Gaussian (var=2.0)
AlignAttention-shared (W+F+G)	1	128	300	100	Gaussian (var=2.0)
AlignAttention-sep (F+G)	1	128	300	100	Gaussian (var=2.0)
AlignAttention-sep (F+W)	1	128	300	100	Gaussian (var=5.0)
AlignAttention-sep (W+G)	1	128	100	50	Gaussian (var=2.0)
AlignAttention-sep (W+F+G)	1	128	300	100	Gaussian (var=2.0)

Table A.1: Hyper-parameters for our best models in IMDB experiments.

Model	layers	d_{lstm}	$seqlen$	noise
LSTMClassify (F+G)	1	300	40	N/A
LSTMClassify (F+W)	1	300	40	N/A
LSTMClassify (W+G)	1	300	40	N/A
LSTMClassify (W+F+G)	1	300	40	N/A
AlignEmbed-shared (F+G)	1	50	40	Gaussian (var=2.0)
AlignEmbed-shared (F+W)	1	150	40	Gaussian (var=2.0)
AlignEmbed-shared (W+G)	1	100	40	Gaussian (var=2.0)
AlignEmbed-shared (W+F+G)	1	150	40	Gaussian (var=2.0)
AlignEmbed-sep (F+G)	1	150	40	Gaussian (var=2.0)
AlignEmbed-sep (F+W)	1	150	40	Gaussian (var=2.0)
AlignEmbed-sep (W+G)	1	150	40	Gaussian (var=2.0)
AlignEmbed-sep (W+F+G)	1	150	40	Gaussian (var=2.0)
AlignAttention-shared (F+G)	1	100	40	Gaussian (var=2.0)
AlignAttention-shared (F+W)	1	150	40	Gaussian (var=2.0)
AlignAttention-shared (W+G)	1	100	40	Gaussian (var=2.0)
AlignAttention-shared (W+F+G)	1	100	40	Gaussian (var=2.0)
AlignAttention-sep (F+G)	1	50	40	Gaussian (var=2.0)
AlignAttention-sep (F+W)	1	150	40	Gaussian (var=2.0)
AlignAttention-sep (W+G)	1	150	40	Gaussian (var=2.0)
AlignAttention-sep (W+F+G)	1	150	40	Gaussian (var=2.0)

Table A.2: Hyper-parameters for our best models in SNLI experiments.