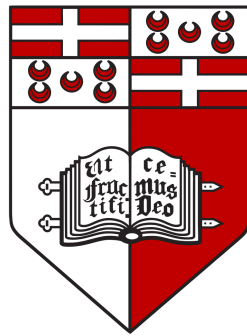


Grounding Natural Language Inference on Images

VU Trong Hoa

MSC. DISSERTATION



Department of Intelligent Computer Systems
Faculty of Information and Communication Technology
University of Malta

Supervisor:
Dr. Albert GATT
Institute of Linguistics and Language Technology

*Submitted in partial fulfilment of the requirements for the Degree of
Master of Science in Human Language Science and Technology*

January 2, 2018

M.Sc. (HLST)
FACULTY OF INFORMATION AND
COMMUNICATION TECHNOLOGY
UNIVERSITY OF MALTA

Declaration

Plagiarism is defined as “the unacknowledged use, as one’s own work, of work of another person, whether or not such work has been published” (Regulations Governing Conduct at Examinations, 1997, Regulation 1 (viii), University of Malta).

I, VU Trong Hoa, the undersigned, declare that this thesis titled, “Grounding Natural Language Inference on Images” and the work presented in it are my own, except where acknowledged and referenced.

I understand that the penalties for making a false declaration may include, but are not limited to, loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree program.

Student Name: VU Trong Hoa
Course Code: CSA5310 HLST Dissertation
Title of work: Grounding Natural Language Inference on Images

Signed:

Date: January 2, 2018

University of Malta

Abstract

Faculty of Information and Communication Technology

Department of Intelligent Computer Systems

Master of Science in Human Language Science and Technology

Thesis: Grounding Natural Language Inference on Images

by Vu Trong Hoa

Despite the surge of research interest in problems involving linguistic and visual information, exploring multimodal data for Natural Language Inference remains untouched. Natural language Inference, regarded as the basic step towards Natural Language Understanding, is extremely challenging due to the natural complex of human languages. However, we believe this issue can be alleviated by using multimodal data. Given an image and its description, our proposed task is to determine whether a natural language hypothesis contradicts, entails or is neutral to the image and its description. To address this problem, we develop a multimodal framework based on the Bilateral Multi-perspective Matching framework. Data is collected by mapping the SNLI dataset with the image dataset Flickr30k. The result dataset, made publicly available, has more than 565k instances. Experiments on this dataset show that the multimodal model outperforms the textual model.

Contents

Declaration of Authorship	iii
Abstract	v
Contents	vii
1 Introduction	1
1.1 Goal and Contribution	2
1.2 The Multimodal Natural Language Inference Task	2
1.3 Related works	3
1.4 Thesis structure	4
2 Background on Neural Networks	7
2.1 Feed-forward Neural Network	7
2.1.1 Logistic Regression	7
2.1.2 Feed-forward Neural Network	8
2.1.3 Error Backpropagation	10
2.2 Convolutional Neural Networks	12
2.3 Recurrent Neural Network	14
2.4 Long Short-Term Memory	15
2.5 Examples of applications in NLP	15
3 Methodologies	19
3.1 Bilateral Multi-Perspective Matching	19
3.1.1 Textual Matching Operations	21
3.1.2 Matching Strategies	22
3.2 Visual Bilateral Multi-perspective Matching	23
3.2.1 Framework overview	23
3.2.2 Multi-modal Multi-perspective Operation	25
3.3 Conclusion	25
4 Experiment results	27
4.1 Dataset	27
4.2 Hyperparameters	28
4.3 Results	29
4.4 Analysis	31
4.5 Summary	31
5 Conclusions and Future work	33
5.1 Conclusion	33
5.2 Contribution	33
5.3 Future work	34

Bibliography	35
List of Figures	39
List of Tables	41
List of Abbreviations	43

Chapter 1

Introduction

Natural Language Inference (NLI) or *Recognizing Textual Entailment* (RTE) is the problem of determining whether a natural language hypothesis h can be inferred from a natural language premise p (entailment) or contradiction or neutral. This problem is extremely challenging due to the complexity of natural languages. However, we believe that the issue can be alleviated by using multimodal data.

Recognizing relation between sentences is difficult but if it is solved, would bring crucial benefits for many practical applications. For example, a question answering system could use an NLI system to examine whether a target question can be inferred from some candidate answers. A machine translation system can use NLI as an evaluation component to measure the semantic equivalence between generated candidates and gold-standard translations, instead of using BLEU score which works on lexical surface. Recently, NLI also become a crucial testing ground for development of distributional semantic representation (Nangia et al., 2017). MacCartney (2009) regarded NLI as a necessary step towards true natural language understanding.

Neural networks are the current dominant methods for NLI (Bowman et al., 2016; Wang et al., 2017; Chen et al., 2017; Nangia et al., 2017). First, these methods are effective in modeling semantic representations, sentence structures. In fact, a simple multilayer perceptron can perform on par with traditional methods (Bowman et al., 2015) and later developed models perform remarkably better. Second, neural networks usually make use of a pre-trained word embedding vectors which play a role similar to that of the knowledge base in a symbolic logic system. The difference is these vectors are trained from the vast amount of text from the Web which makes it less expensive and domain-restricted. Finally, it is feasible to train a joint multimodal model using neural networks as they are all represented as vectors and matrices.

Recently, as a result, there is a great interest among the Natural Language Processing community and Computer Vision community in problems that requiring linguistic and visual data. The recent years have seen a surge in multimodal data with billions of videos and image posted everyday. In fact, videos and images even become dominant modality as in Vine, Instagram or Youtube. Interestingly, in science, research also suggest that meaning of words and sentences in natural language should be grounded in physical objects and actions (Roy, 2005; Andrews et al., 2009).

In light of previous research, in this thesis, we are interested in examine the multimodal settings for NLI. We would like to see if visual information could or should be used in order to boost performance of an NLI system.

1.1 Goal and Contribution

The purpose of this thesis is to explore an effective method to exploit visual information to improve the NLI task. The challenge is to make an inference from an image or how to utilize both modalities in a model. To this end, we extend the textual BiMPM framework (Wang et al., 2017) to work with images. Our proposed multimodal model outperforms the textual BiMPM.

We form a dataset for this purpose from SNLI and Flickr30k. The dataset contains more than 565k instances. We make this dataset publicly available together with the implementation.

1.2 The Mutimodal Natural Language Inference Task

We propose an extension of NLI called Mutimodal Natural Language Inference Task. Given an image I and its description as a premise p , the task is to determine whether a hypothesis h entails, contradicts or is neutral with respect to the premise, where this process can be supported by information from the image.

This definition, as many other definition in NLP, refers to human understanding of languages, ability to observe, and assumes commonsense knowledge, on which inference judgment relies. A simple example bellow explains the task better:

- (i) The image I:



- (ii) Premise P : A black cat lying on a bench.
- (iii) Hypothesis H_1 : 2 red cats are playing on a bench.
- (iv) Hypothesis H_2 : There is a cat on a bench
- (v) Hypothesis H_3 : The cat is waiting for her owner.

In NLI settings, given the premise and the image I , H_1 is considered to be contradicted, H_2 is a proper inference and H_3 is neutral. The reason for the selection of those labels is that a person who observes the image and premise would plausibly come to the same conclusion.

In this task, the image plays the role of a grounding source. For example, a normal human can easily draw the same labels for those three hypothesis with regards to only the image. Indeed, for this simple task, it is probably easier, more intuitive for humans to work with visual data. However, whether computers can have a similar ability remains a question.

Assume we have an ideal matching scheme between two modalities, visual information can help the inference process. First, there is less potential ambiguities in visual scenes than in natural languages. The fact that natural languages can have many linguistic phenomena makes it harder to deal with. For example, Lynch et al. (2016) found that, in the automatic ranking search results, thumbnails seem to be less “noise” and multimodal data are more reliable. Second, information on the image can be verified by the text and vice versa. This helps to strengthen the inference process. Third, the image is a richer input for the inference task. For instance, we can conclude that the cat is outdoors and that the bench is brown, but neither of these pieces of information is in the textual premise.

It is important to notice that NLI works with *directional* relationships which means if the A entails B , the reversed direction does not hold. If it does, the problem becomes *paraphrase identification* which can be solved by running an NLI system with input A, B and B, A .

Also, NLI can be seen as a 3-class classification problem. Given a pair $x = \langle I, P, H \rangle$, the classifier is asked to determine whether its label y belongs to one of there class: *entailment*, *neutral*, and *contradiction*.

1.3 Related works

Early successful methods heavily rely on lexical surface representation and syntactic similarity which is inspired from information retrieval and machine translation. In general, the possibility of entailment is determined based on lexical overlap between the premise and the hypothesis. Obviously the systems could not require all the text to be matched, they usually train to get a threshold of the proportion of the text to be able to predict (Jijkoun and Rijke, 2005). Geffet and Dagan (2005) extends the above idea by introducing stemming, lemmatizing, lexical resources such as Wordnet and probabilistic mappings produced by distributional similarity approaches to increase the chance of matching. Glickman and Dagan (2005) compares the words of the premise and hypothesis using the results from an Internet search engine. Their system achieved the best performance in the first PASCAL RTE Challenge (Dagan et al. (2006)).

For semantic researchers, applying formal logic is the most intuitive answer for NLI. This is done by translating natural language into logical formulas then use logical inference to determine the relation (Akhmatova, 2005; Bayer et al., 2005; Bos and Markert, 2006). These methods have advantages dealing with negations, quantifiers, conditions. However, they are domain-restricted. The reason as discussed in MacCartney (2009), is because it is challenging to handle natural language phenomena

including idioms, ellipsis, anaphora, paraphrase, ambiguity, vagueness, aspect, lexical semantics, the impact of pragmatics, and so on.

The recent released dataset SNLI by (Bowman et al., 2015) has enabled the research community to apply various neural network architectures and so far many exciting results has been published. The dataset contains 570k pairs of sentences together with their relation is either *entailment*, *neutral* or *contradiction*. The premise is taken from an image corpus called Flickr30k. Before this release, the research community usually relied on datasets from the RTE challenges¹, the Sentences Involving Compositional Knowledge (SICK) from SemEval 2014 task (Marelli et al., 2014). Those dataset are very high quality, manually created but small in size which is not suitable for neural networks methods. The size of SNLI, however, makes it feasible for applying neural networks methods which usually require large amount of data. Indeed, the success of this dataset has been demonstrated in a vast number of studies, some of them even surpass human performance².

While a simple neural network architecture proposed in Bowman et al. (2015) achieve 77.6% on SNLI and approach the state-of-the-art result on SICK dataset, later proposed models are more advanced. Mou et al. (2016) proposed a “sentence encoder” architecture where 2 encoders shared parameters are employed to turn the hypothesis and premise in to two vectors. By using a tree-based convolutional neural network in each encoder, their model obtains an accuracy of 82.1% on the SNLI dataset. These vector are then concatenated with their element-wise product, sum, or difference. Figure 1.1 illustrates this architecture. This architecture becomes common for testing sentence representation (Nangia et al., 2017).

Wang et al. (2017) propose a far more elegant attention-based matching framework. In the framework, each context-aware token of either premise or hypothesis is matched with the tokens in other sentence using various weighting strategy. The multi-perspective matching operation is an extension of cosine similarity where inputs are projected to different spaces and each output value is cosine similarity in each space. The single BiMPM yields 86.9% on SNLI. In this thesis, we make use of this framework to evaluate the effectiveness of visual information.

Previous research in NLP and CV has showed benefit of using multi-modal data over text-only modal. Specia et al. (2016) show improvements in translating image captions by using visual information from the images. Pasunuru and Bansal (2017) found that, in a multitask setting, video information can help the entailment generation task and entailment generation task helps improve video captioning.

1.4 Thesis structure

Chapter 2 gives the necessary basic models that are required for building the proposed models and delivers all important definitions. Section 2.1 explains how to build a feed-forward network, the basic component of neural networks. Based on this component, we present some successful models in Computer Vision and Natural Language Technology in Section 2.2 and Section 2.4. These models are the main modules in our proposed models.

¹<https://tac.nist.gov/2011/RTE/>

²Statistics of performance on the dataset are available on <https://nlp.stanford.edu/projects/snli>

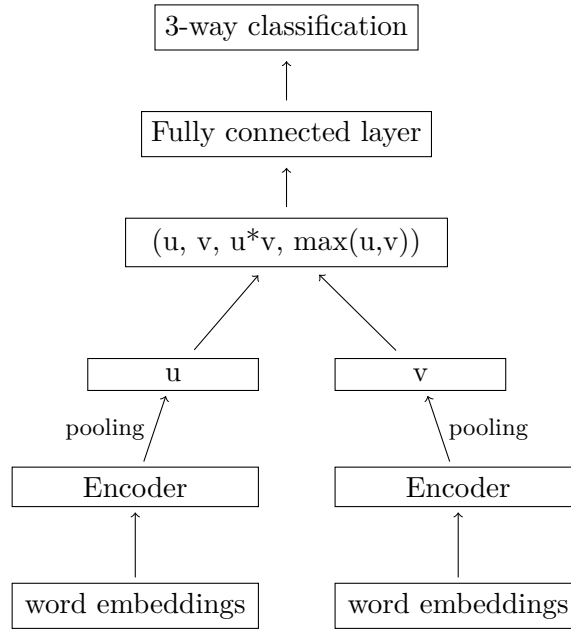


FIGURE 1.1: A "sentence encoder" architecture.

Chapter 3 presents the methodologies we use to demonstrate the effectiveness of multimodal data. The textual framework BiMPM is presented in Section 3.1. Our proposed multimodal framework is presented in Section 3.2

Chapter 4 presents our experiments with our proposed model on SNLI dataset. We first present our data collection procedure, experiment results then give some insights into how the model performs.

Chapter 5 gives a conclusion of the thesis and presents possible improvements that could be made.

Chapter 2

Background on Neural Networks

The first development of neural networks dates back in 1943 when McCulloch and Pitts developed a linear model that could test if output of $f(x, w)$ is positive or negative. This was followed by the perceptron (Rosenblatt, 1958) which is the first model that can learn weights from data. After a long history with many efforts and inventions, neural networks made a recent comeback as Hinton et al. (2006) and Bengio et al. (2007) came up with a strategy to train a deep neural network. Since then, the term neural networks aka *deep learning* started to become common in different communities studying different tasks including natural language processing, vision, motion planning and speech recognition.

In this chapter, we would like to present the necessary basic models that are required for building the proposed models and delivers all important definitions. We first start with the basic Feed-forward networks in Section 2.1 then we describe some successful models in Computer Vision and Natural Language Technology, namely Convolutional Neural Network in Section 2.2 and Long Short-Term Memory in Section 2.4. In the end of this chapter, we introduce some common usage of neural networks in NLP which also appear in our proposed model.

2.1 Feed-forward Neural Network

2.1.1 Logistic Regression

Logistic Regression for multiclass classification:

$$y(x, w) = f\left(\sum_{j=1}^M w_j \phi_j(x)\right) \quad (2.1)$$

Where w_j are trainable parameters. $\phi(x)$ are feature functions, one of these is set to constant, say $\phi_0(x) = 1$ so that w_0 is a bias. $f(\cdot)$ is a softmax function, a nonlinear activation function turns numbers into proper probabilities. The posterior class probabilities are given by

$$p(C_k|\phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \quad (2.2)$$

where the 'activations' a_k are given by

$$a_k = w_k \phi \quad (2.3)$$

To train this model, we use maximum likelihood to adjust the parameters w . Let $\{\phi_n, T\}$ be the data set where $\phi_n = \phi(x_n)$ and T is the $N \times K$ matrix of one-hot target variables, each variable is a vector with all elements zero except the k^{th} value for the target class \mathcal{C}_k . $y_{nk} = y_k(\phi_n)$. The likelihood function given by:

$$p(T|w_1, \dots, w_K) = \prod_{n=1}^N \prod_{k=1}^K p(\mathcal{C}_k|\phi)^{t_{nk}} = \prod_{n=1}^N \prod_{k=1}^K y_{nk}^{t_{nk}} \quad (2.4)$$

We take the negative logarithm:

$$E(w_1, \dots, w_K) = -\ln p(T|w_1, \dots, w_K) = -\sum_{n=1}^N \sum_{k=1}^K t_{nk} \ln y_{nk} \quad (2.5)$$

which is the *cross entropy* cost function for multiclass classification.

Now, to minimize this cost, we take the gradient of the function regarding one parameter w_j . First, we calculate the derivatives of y_k (result of softmax function) with respect to the activations:

$$\frac{\partial y_k}{\partial a_j} = y_k(I_{kj} - y_j) \quad (2.6)$$

where I_{kj} are elements of the identity matrix. Making use of equation 2.6 and $\sum_k t_{nk} = 1$, we obtain

$$\nabla_{w_j} E(w_1, \dots, w_K) = \sum_{n=1}^N (y_{nj} - t_{nj}) \phi_n \quad (2.7)$$

As we see in the Equation 2.7, the contribution for the gradient from data point n is given by the loss $y_{nj} - t_{nj}$ between the target value and the prediction value, times the feature function. For the sake of simplicity, we could formulate a on-line algorithm in which data are presented one at a time and the w_j is updated using

$$w^{(t+1)} = w^{(t)} - \eta \nabla_{w_j} E \quad (2.8)$$

in which η is the learning rate.

Linear models are simple, stable and easy to train however their modeling ability is limited. For a M -dimensional feature space, this model requires only M trainable parameters. Linear operations are relatively cheap to compute. They usually are optimized in to matrix multiplications and are executed really fast using GPUs. However, linear models can only model linear relations which not always happen in practical applications. A win-win solution is to introduce non-linearities into these models. The way to do that is to stack multiple linear models together with non-linear functions and what we got are feed-forward neural networks. Logistic regression model itself is a one-layer feed-forward neural network.

2.1.2 Feed-forward Neural Network

As discussed above, the idea of these models is extended from logistic regression which take the form 2.1 to represent more complex relations. Our goal is to extend the feature functions $\phi_j(x)$ to depend on parameters which can be trained along with

w . There are many ways to achieve this including using the function following the same form 2.1, that makes each feature function becomes a nonlinear combination of inputs with trainable parameters. The *activations* a_k in Equation 2.3 now are written as:

$$a_k = \sum_{j=1}^M w_{kj}^{(2)} h_j + w_{k0} \quad (2.9)$$

$$h_j = g\left(\sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}\right) \quad (2.10)$$

where g is a nonlinear activation function which computes *hidden units* h_j from inputs. K , D , M are number of outputs, hidden units and inputs respectively. w_{kj} and w_{ji} are trainable parameters. w_{k0} and w_{j0} are biases.

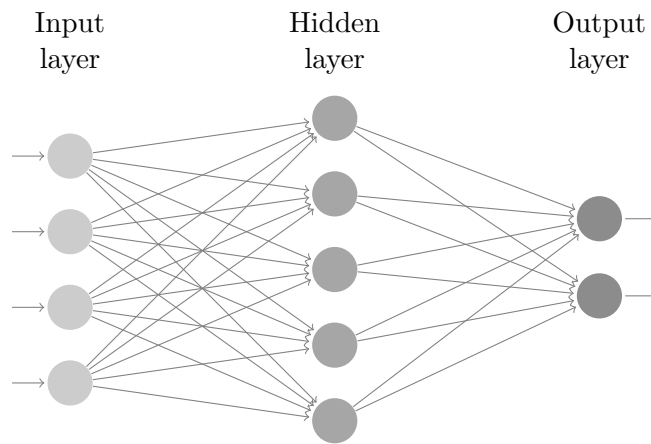


FIGURE 2.1: An example of a 2-layer feed-forward network where inputs, hidden units, output are denoted by nodes and weights are represented by arrows

The model above is called 2-layer feed-forward network(FFN), illustrated in Figure 2.1. The information flows through the network in the forward direction, from the input layer, through a hidden layer and finally to the output layer. There are no feedback connections connecting later stages back to the input. We will discuss some networks having feedback connection in Section 2.4 including Recurrent Neural Network and Long Short-Term Memory.

These models are called networks as the function they represent is a composition of other functions, that can further be decomposed into other functions. This can be conveniently illustrated as a network of chain structure. The first function transforming the inputs is the *first layer*. The final layer is called *output layer*. The depth of the model is defined as the number of the computation layers, input layer is not included. This is where the term “deep learning” comes from. Although, “deep learning” can also be referred as methods for learning feature representation at successively higher, more abstract layers(Deng and Yu, 2014). Values at the middle layers are not known for the training data and the training process needs to make use of these layers together with the output layer to give desired output. These layers are called *hidden layers*, each of them contains many *hidden units*.

The term *neural* comes from the fact that these models are loosely inspired by the biological nature of the human brain. Each *unit* in a layer play a similar role as a

neuron as it also receives input from other units and computes its own activation value. However, Goodfellow et al. (2016) points out the fact that neural networks nowadays are guided by many mathematics and engineering and it is better to treat feed-forward network as function approximation machines rather than as models of brain functions.

The simplest way to train this model is using gradient information to adjust the weights which take a small step in the direction of negative gradient:

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E(w^{(\tau)}) \quad (2.11)$$

in which $\eta > 0$ is the learning rate. After each update, the weights are adjusted to the direction of decrease the error function E and this method is known as *Gradient Descent*. Although it is very intuitive, it requires the whole training set to be processed at a time. The calculation of gradient is discussed in the section 2.1.3.

Goodfellow et al. (2016) highlight the difference between training a linear model and a feed-forward network is that nonlinearity turns most common loss functions to become non-convex. This means it is not guaranteed that applying stochastic gradient descent to these functions will find a convergence. It is advised to initialize all weights to small random values and biases to either zero or small positive values. Stochastic gradient descent, in contrast to the on-line algorithm we mention in Section 2.1.1, updates parameters after seeing a few training examples instead of the full training set.

Despite the fact that feed-forward networks are straightforward, its computation ability is nontrivial. The universal approximation theorem (Hornik, 1991) claims that a standard feed-forward network with a single hidden layer that contains finite number of hidden units and with arbitrary activation function can approximate any Borel measurable function with any desired non-zero amount of error.

The universal approximation theorem means that neural networks can learn arbitrary functions; however, it does not guarantee the training algorithm will be able to find the right parameters or will not suffer from overfitting. The theorem also does not specify how large the network should be and in the worst case an exponential number of outputs may be required (Barron, 1993). However, using deeper models would help to reduce number of units needed and reduce the errors. This explains how the term “deep learning” is getting more and more popular.

2.1.3 Error Backpropagation

As we already discussed in the previous section, we can optimize weight parameters by making step by step toward the direction of negative gradient of the error function. We also knew the gradient of error function at the output layer but we do not know the gradient with regard to the weights at the hidden layers. The solution is using *error backpropagation* or simply *backprop* which alternately passes information forward and backward through the network. The formula is illustrated on a 2-layer FFN together with a sum of squares error function but can be applied to many other kinds of network, not just the FFN.

The backprop algorithm has two stages: forward and backward. In the forward stage, we apply input vector x_n to the network, forward it through the network to find the

activation values of hidden units and output units using the Equation 2.9 and 2.10. Specifically, at the first layer:

$$y_k = \sum_i w_{ki} x_i \quad (2.12)$$

And at the hidden layers, activations are calculated by

$$a_j = \sum_i w_{ji} z_i \quad (2.13)$$

$$z_j = h(a_j) \quad (2.14)$$

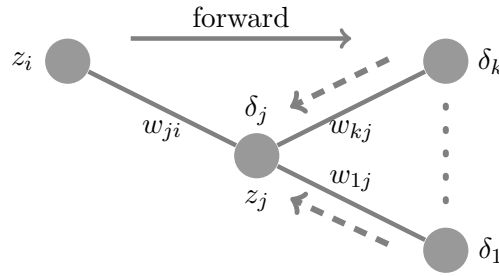


FIGURE 2.2: Calculate error δ_j at hidden unit j by backpropagate (dash arrows) errors from all the nodes that j sends information

In the backward stage, we need to calculate the derivatives of the error function with respect to each of the weights. Let say we train the network using maximum likelihood and the error function is sum of square error of each data point:

$$E(w) = \sum_{n=1}^N E_n(w) \quad (2.15)$$

$$E_n = \frac{1}{2} \sum_k (y_{nk} - t_{nk})^2 \quad (2.16)$$

where $y_{nk} = y_k(x_n, w)$ is output at data point n . Now let consider the derivative of E_n with respect to a weight w_{ij} . We notice that E_n depends on the weight w_{ij} only by the sum a_j given by Equation 2.13. Apply chain rule, we get the derivative:

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \quad (2.17)$$

Using 2.13 we have:

$$\frac{\partial a_j}{\partial w_{ji}} = z_i \quad (2.18)$$

Let

$$\delta_j = \frac{\partial E_n}{\partial a_j} \quad (2.19)$$

By substituting 2.18 and 2.19 to 2.17, we have

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i \quad (2.20)$$

Equation 2.20 tells us that we only need to calculate the error δ_j in order to obtain the required derivative. For the output unit, we have the error for each class:

$$\delta_k = \frac{\partial E_n}{\partial a_k} = y_k - t_k \quad (2.21)$$

For hidden units, let use the chain rule again:

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j} \quad (2.22)$$

where the sum is over all the k units that the current unit sent connections. The backward process is shown in the figure 2.2. The units k could be either hidden units or output units. By substituting 2.19 into 2.22 and making use of 2.13 and 2.14, we have:

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k \quad (2.23)$$

This is the *backpropagation* formula which tells us to calculate the error backward from the later units in the network and because we already know the error value for the output, all the error δ can be calculated recursively.

2.2 Convolutional Neural Networks

Covnets are a specialized kind of network for processing data that has a grid-like topology, for example 2D grid of image pixels. Convolutional neural networks (Cov-Nets) were first developed by LeCun et al. (1989) but were not popular due to hardware limitations. In the visual recognition challenge ILSVRC 2012¹, CovNets made a comeback by achieving a top-5 error rate of 16% while best methods based on standard vision features achieved achieve 26% (Krizhevsky et al., 2012). In 2015, ResNet (He et al., 2016) another CovNet architecture reduced this rate to 3.57%.

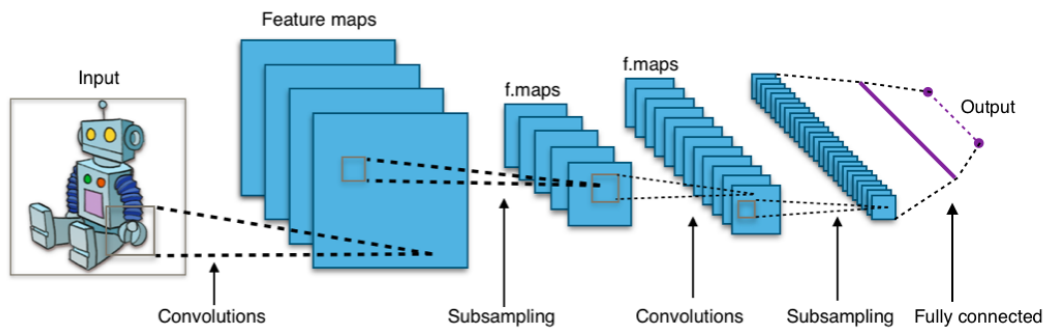


FIGURE 2.3: A typical convolutional neural network architecture²

There are many variations in CovNets architectures but are built from three main types of layers: *convolutional layer*, *pooling layer*, and *fully-connected layer*. Figure 2.3 shows an example of a typical CovNets architecture. This network contains a *convolutional layer* interacting with the input, then followed by a *pooling layer*. The network goes on with another *convolutional layer* and a pooling layer before transform

¹<http://www.image-net.org/challenges/LSVRC/>

²https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Typical_cnn.png

to output by a *fully-connected layer*.

Convolutional Layer: This is the most important part of CovNets and its name is inspired by the “convolution operations” which are performed in this layer. Figure 2.4 illustrates an example of a convolution operation. There is a set of *filters* (or *kernels*) sliding along the image width and height. For example, a filter may be a tensor size $4 \times 4 \times 3$ which is 4 pixels width, 4 pixels height and depth of 3 (3 channels red, blue and green of images). The value of the filter tensor is trainable. Every time we slide the filter, we compute the dot product between the filter tensor and the input area covered by the filter:

$$(\mathbf{I} * \mathbf{K})_{r,s} = \sum_{u=-h_1}^{h_1} \sum_{v=-h_2}^{h_2} \mathbf{K}_{u,v} \mathbf{I}_{r+u,s+v} \quad (2.24)$$

where K is a filter tensor, (r, s) is the current point on the input I . h_1, h_2 are size of the filter. The result of this process is a 2D feature map (see figure 2.3). As we apply a set of filter to the input, the output after this layer is a set of activation maps (see figure 2.3 and 2.4).

Pooling Layer (also called subsampling or downsampling): The purpose of this

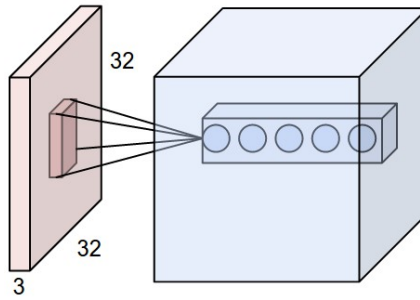


FIGURE 2.4: An example of a convolutional layer. The input image having size 32×32 with 3 color channels. After applying a set of filter, each filter return a 2D feature map, we obtain a 3D output tensor.

layer is to gradually reduce the size of the input and reduce number of parameters which help to optimize amount of computation and control overfitting. Polling layers are usually used between successive convolutional layers (See figure 2.3). Pooling layer can implement various types: *max*, *average*, *sum*. The most common form of is a pooling layer with filters having size 2×2 . Every time we slide this filter, we take a sample of the region covered by the filter. For example, in case of max pooling and 2×2 filter, we take a max over 4 numbers. The depth dimension of the feature maps remains unchanged.

Fully-connected Layer: As discussed in previous section 2.1.2, hidden units in this layer are fully connected to all activations in the previous layer. That is also the only difference between this layer and the convolutional layers. In convolutional layer, each part of the outputs only connect with a region in the input and many parts of the output share parameters.

Zeiler and Fergus (2014) explained why Covnets have high performance on many computer vision tasks by visualizing feature maps of different layers of the fully-trained models and training models. First, they found out that each layer of the network learning different types of shapes and objects and greater invariance at higher layers. Second, the networks are robust to feature variance. The network output is stable to translations and scalings.

VGGnet (Simonyan and Zisserman, 2014) is one of the common Convnets architecture, the runner-up in ILSVRC 2014. This architecture shows us that the depth of the network is a crucial component for good performance. Their pre-trained models are widely used in computer vision research community as it is possible to use these models to extract features from images at different level of abstraction.

2.3 Recurrent Neural Network

If ConvNets are the dominant method in Computer Vision, Recurrent Neural Networks (RNN) play the similar role in Natural Language Processing. RNNs are family of neural network architectures for dealing with sequential data. Given an ordered list of input vectors, at each point in time, RNNs process the input vector with regards to the output of the previous state. We can represent RNNs as deterministic transitions from previous to current hidden states.

$$RNN : x_t, h_{t-1} \rightarrow h_t \quad (2.25)$$

Where x_t is the input at time t , h is the transition function. If we consider a vanilla RNN, h is given by

$$h_t = f(Wx_t + Uh_{t-1}); \quad f \in \{sigmoid, tanh\} \quad (2.26)$$

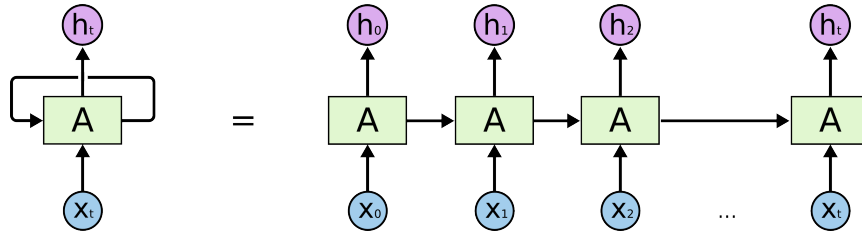


FIGURE 2.5: A recurrent neural network unrolled through time.

In comparison to feed-forward networks, RNNs take advantages of sharing parameters. For example, Figure 2.5 shows an unfold form of RNN. At different points in time, the network may process different inputs but the parameters remain the same. As a result, the models have much less parameters, allows the models to handle long sequences.

Backpropagation Through Time To train RNNs, we can use the Error Backpropagation discussed in 2.1.3 but on the model's unfold form through time (Figure 2.5). In short, the Backpropagation Through Time (BPTT) first unfolds the network then calculate the errors at each time step. After rolling back the network, BPTT update the weights with the accumulated error. This is where the *gradient vanishing* can happen as the weights can get close to zero after a long sequence of accumulation operations. Long short-term memory, discussed next section, is designed to solve this issue.

2.4 Long Short-Term Memory

Long short-term memory (LSTM), proposed by Hochreiter and Schmidhuber (1997), is one of the recurrent neural networks processing sequence data. For a long sequence, there is a well-known issue called "vanishing gradient". LSTM address this issue by using few gate vectors and memory that "memorize" information. LSTM has *input*, *forget* and *output gates* helping it to decide to overwrite the memory cell, retrieve it, or keep it for the next time step. This allows LSTM to preserve long-range dependencies while process long sequences. LSTM can be described using deterministic transitions from previous to current hidden states:

$$LSTM : h_t^{l-1}, h_{t-1}^l, c_{t-1}^l \rightarrow h_t^l, c_t^l \quad (2.27)$$

Let $T_{n,m} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be an affine transform ($Wx + b$ for some W and b)

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh} \end{pmatrix} T_{2n,4n} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \quad (2.28)$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g \quad (2.29)$$

$$h_t^l = o \odot \text{tanh}(c_t^l) \quad (2.30)$$

Here, i, f, c, o, g are respectively input, forget, memory, output and hidden state of LSTM. While $\text{sigm}, \text{tanh}, \odot$ are the logistic sigmoid, tanh activation and element-wise multiplication functions respectively.

Bidirectional LSTM (BiLSTM) (Graves and Schmidhuber, 2005) is an extension of LSTM. As we see in equation 2.27, LSTM takes previous hidden states into account while processing current input. BiLSTM, on the other hand, read the input sequence from left to right and from right to left by employing two separate LSTMs. The output of BiLSTM at each point in time is the concatenation of outputs by the two LSTMs.

2.5 Examples of applications in NLP

In NLP, as Goldberg (2016) pointed out, RNNs are the dominant architecture and their most successful type of RNN architecture LSTMs are responsible for many state-of-the-art sequence modeling results. There are various uses of RNN in NLP including *accepter*, *encoder*, *transducer* and *encoder-decoder*.

Acceptor: First, RNNs can be used as an *accepter* where output of the model is decided on the final hidden state (Figure 2.6). For example, we could use an RNN to read characters of a word as an ordered sequence and use the last state to predict the part-of-speech of that word (Ling et al., 2015), or use RNN to read words in a sentence and predict the polarity of the sentence by the last hidden state.

Encoder: Again, we can make use of the last hidden state or concatenation of all hidden states, treat it as the encoding of the sequence. Unlike the *accepter*, decision

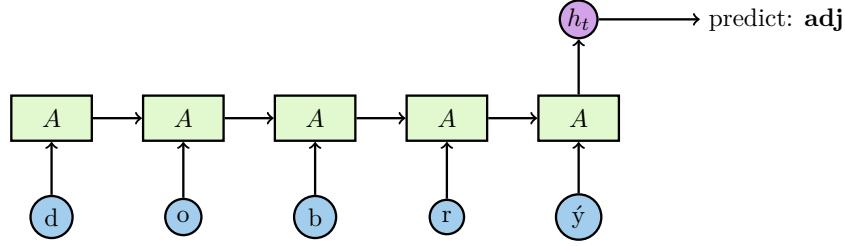


FIGURE 2.6: Example of Acceptor RNNs

is not made directly on this state but instead use it as additional information together with other signals. For example, Bowman et al. (2015) concatenate encoding of two sentences obtained by an LSTM and put it into a feed-forward network to classify the relationship of these sentences. Figure 1.1 illustrates this architecture. In our proposed model, the use of RNN falls into this type.

Transducer: Third, we could use RNN as a transducer, giving an output for every input. An example for this architecture is a sequence tagger, where we use the hidden states to predict the tag at the corresponding positions. A CCG super-tagger by Xu et al. (2015b) using this architecture produce state-of-the-art results on CCG super-tagging. Another example for transduction setup is language models where previous words are used to predict a distribution over the next word. RNN-based language models by Mikolov et al. (2010), Sundermeyer et al. (2012), and Mikolov (2012) outperform traditional methods.

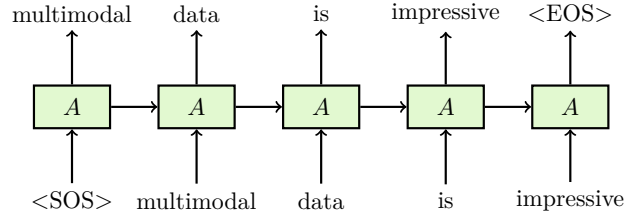
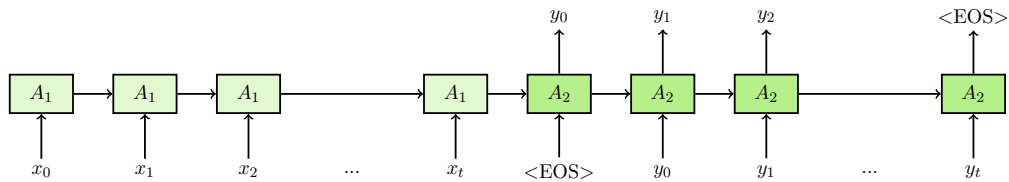


FIGURE 2.7: An example of RNN-based language model working as a transducer

Encoder - Decoder: This can be seen as a special case of *encoder* as, in fact, it employs one RNN to encode the sequence into a fixed-length vector. This vector is used as input to the other decoder RNN. In machine translation, for example, Sutskever et al. (2014) employ an LSTM to encode a variable-length source sentence to a fixed length vector and employ another LSTM to decode the vector. The later LSTM works as a *transducer*.

FIGURE 2.8: An encoder-decoder translating sequence " $x_0x_1x_2...x_t$ " to sequence " $y_0y_1y_2...y_t$ "

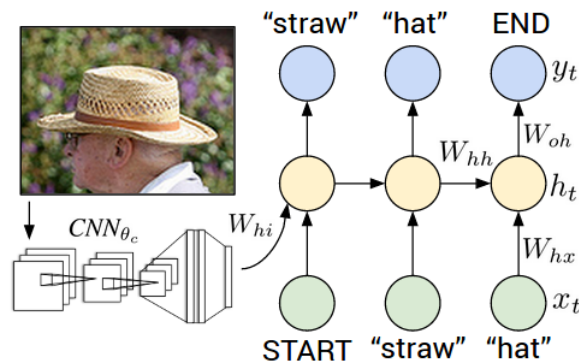


FIGURE 2.9: An example of image captioning system from Karpathy and Fei-Fei (2015)

Thank to the development of neural networks, the Computer Vision and Natural Language Processing are moving closer (Bernardi et al., 2016). In fact, *image captioning* and *visual question answering* has become a key task in both communities. It is common that there are a CovNet to get the visual information and an RNN to process the textual data. Our proposed model has a similar architecture. Figure 2.9 demonstrates an example architecture to produce descriptions from images. The fixed-length vector extracted by the Covnet are used as the initial state for RNN. The generated descriptions are surprisingly realistic (Karpathy and Fei-Fei, 2015).

To this point, we already introduce all necessary information to build and train a neural network as well as how they process text and image. In the next chapter, we make use of the models and definitions given here to describe our proposed model.

Chapter 3

Methodologies

Following the success of neural networks in NLI and multimodal tasks, we opt to use neural networks as our approach. First, we choose BiMPM framework as a baseline model. This framework is reported to perform well on many tasks including NLI. Next, we incorporate visual information into the framework while keeping other parts as-is. We examine different ways of how the hypothesis, the premise and the image interact with each other. Performance of each variant is reported in the next chapter.

Section 3.1 describes the BiMPM framework proposed by Wang et al. (2017). This framework works with text only. Our multimodal framework is described in Section 3.2.

3.1 Bilateral Multi-Perspective Matching

Wang et al. (2017) propose the Bilateral Multi-Perspective Matching framework (BiMPM) to solve various tasks in natural language matching including NLI. Figure 3.1 shows the overview of the architecture. Their ensemble models achieve the best result ¹ on the SNLI dataset at the time of this thesis.

Given a premise (P), a hypothesis (Q), the model predicts whether the hypothesis entails the premise by calculating the probability distribution of the output class $P(y|P, Q)$. The model receives P, Q as inputs and forwards them through five layers to calculate the probability distribution. In the first layer, words are turned into vectors. In the second layer, word vectors are incorporated with its context information and matched with other sentences in the next layers, using different strategies. The aggregation layer turns matching result vectors into fixed-length vectors which are concatenated and fed into a softmax to get the probability distribution. This 5-layers architecture is adopted from Wang et al. (2017).

Embedding layer. This layer not only turns words into vectors but also preserves their semantic relations. Specifically, the framework makes use of the pre-trained words embeddings from Pennington et al. (2014). The motivation behind these kind of embedding is the distributional hypothesis by Harris (1954) which states that words occurring in similar contexts have similar meaning. In fact, word embeddings can capture many linguistic regularities, for example “*paris*” - “*france*” \approx “*rome*” - “*italy*”. The word embedding plays a similar role as the knowledge base in systems using a logic-based approach.

¹Results on SNLI dataset are listed here: <https://nlp.stanford.edu/projects/snli/>

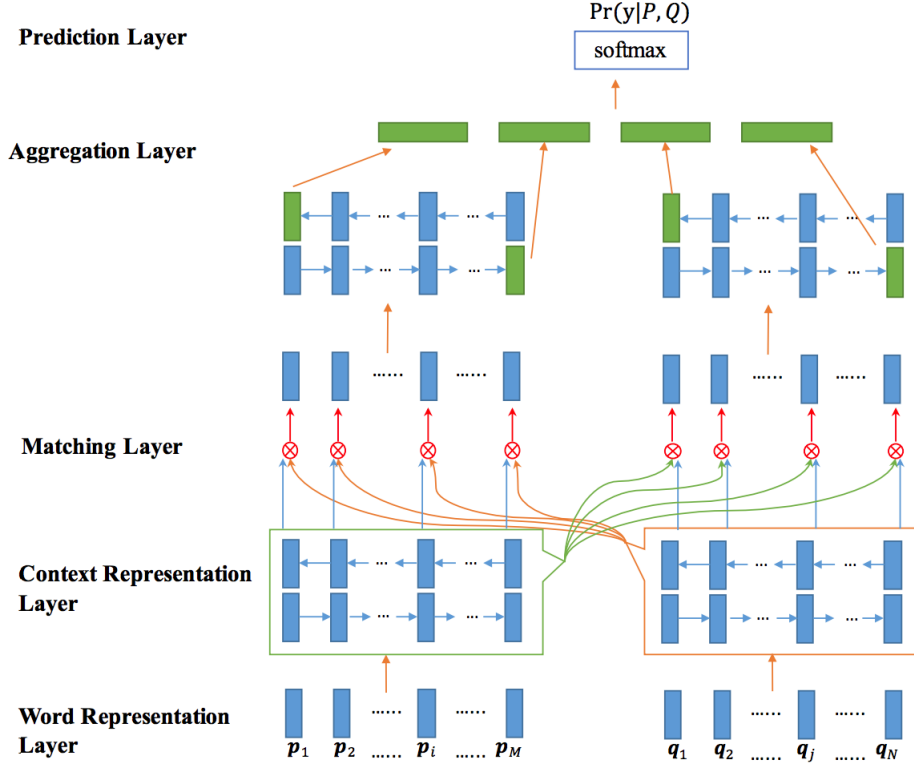


FIGURE 3.1: Overview of BiMPM architecture.

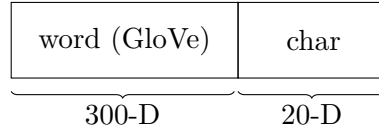


FIGURE 3.2: Word embedding components in BiMPM

To deal with out-of-vocabulary words, the word embedding is supported by the character embedding. The character embedding is obtained by running an LSTM over the sequences of characters and getting the last hidden state. The vector of each character is initialized randomly and trained together with the whole framework. Figure 3.2 shows the components of the embedding. After the first layer, the premise and hypothesis are represented as arrays of word embedding vectors.

$$\begin{aligned}
 \text{Premise } P : & \quad [p_1, \dots, p_M], & p_i \in \mathbb{R}^D \\
 \text{Hypothesis } Q : & \quad [q_1, \dots, q_N], & q_i \in \mathbb{R}^D
 \end{aligned} \tag{3.1}$$

Context layer. The goal of this layer is to incorporate each time step of the premise and hypothesis with its contextual information. To this end, a bi-directional LSTM (BiLSTM) is employed. It receives sequence of embedding vectors as input and outputs corresponding number of output contextual vectors in each direction. The same BiLSTM is used to encode both premise P and hypothesis Q .

$$\begin{aligned}\overrightarrow{h_i^p} &= \overrightarrow{LSTM}(p_i, \overrightarrow{h_{i-1}^p}) \\ \overleftarrow{h_i^p} &= \overleftarrow{LSTM}(p_i, \overleftarrow{h_{i-1}^p})\end{aligned}\tag{3.2}$$

$$\begin{aligned}\overrightarrow{h_i^q} &= \overrightarrow{LSTM}(q_i, \overrightarrow{h_{i-1}^q}) \\ \overleftarrow{h_i^q} &= \overleftarrow{LSTM}(q_i, \overleftarrow{h_{i-1}^q})\end{aligned}\tag{3.3}$$

Matching layer. This is the most crucial part of the BiMPM framework where each contextual embedding (time-step) of a sentence is matched against all contextual embedding of the other sentence. As shown in Figure 3.1, the hypothesis and premise are compared in two directions: compare each time-step of the hypothesis (Q) to all time-steps of the premise (P) and compare each time-step of the premise to all time-steps of the hypothesis. The textual matching operation is denoted by \otimes in Figure 3.1 and is defined in the section 3.1.1 in contrast to the multimodal matching operation defined in section 3.2.2 which was developed in the present thesis.

Aggregation Layer. The purpose of this layer is to obtain a fixed-length vector from the sequence of matching vectors. Another BiLSTM is used to encode those matching sequences separately. The fixed-length vector is then constructed by concatenating the last time step of the outputs from BiLSTM models.

Prediction Layer. This is the last layer in the framework and where probability distribution is calculated. The framework employs a two-layer feed-forward network to transform the fixed-length vector and apply a softmax function to the output vector to obtain a proper probability distribution which indicates probabilities of the label y to be *entailment*, *neutral* or *contradiction*.

3.1.1 Textual Matching Operations

This is similar to cosine similarity except the result of these operations are vectors instead of scalars. We keep it as-is from BiMPM (Wang et al., 2017).

$$m = f_m(v_1, v_2; \mathbf{W})\tag{3.4}$$

Each vector input is replicated l times and multiplied with different weight vectors W_k before obtaining a cosine similarity.

$$m_k = \text{cosine}(W_k \circ v_1, W_k \circ v_2)\tag{3.5}$$

where v_1 and v_2 are d-dimensional vectors. \mathbf{W} is a $l \times d$ trainable weight matrix. m is a l-dimensional matching result vector $m = [m_1, \dots, m_l]$. The idea of this operation is very similar to the idea from self-attentive sentence embedding (Lin et al., 2017) where each sentence has various meaning channels. Each channel encode different part of the sentence.

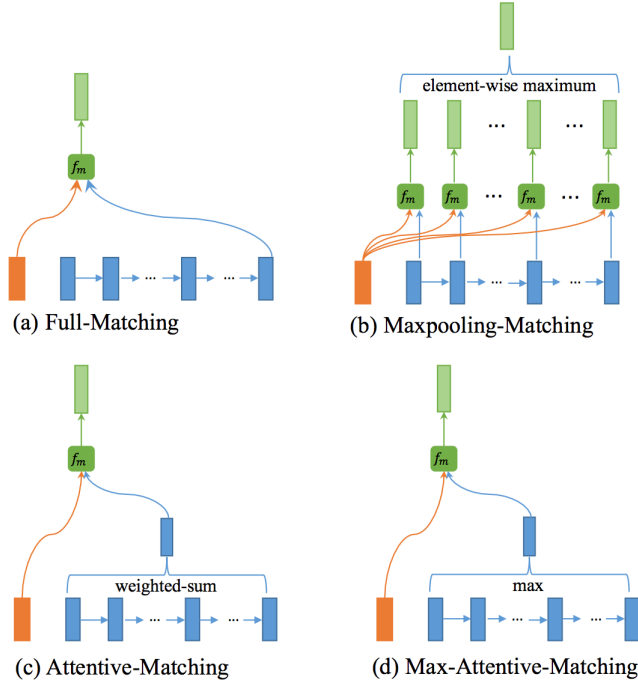


FIGURE 3.3: Diagram for 4 matching strategies

3.1.2 Matching Strategies

As we discussed, in the matching layer, each time-step of one sentence is matched with the other sentence in various strategies. Specifically, the strategy can be *fully-matching*, *maxpooling-matching*, *attentive-matching*, *max-attentive-matching*.

Full-Matching: This strategy is shown in Figure 3.3 (a) where each forward (backward) time-step \vec{h}_i^p (or \overleftarrow{h}_i^p) of P is matched with the last forward (backward) time-step \vec{h}_N^q (or \overleftarrow{h}_1^q) of Q .

$$\begin{aligned}\overrightarrow{m}_i^{full} &= f_m(\vec{h}_i^p, \vec{h}_N^q; \mathbf{W}^1) \\ \overleftarrow{m}_i^{full} &= f_m(\overleftarrow{h}_i^p, \overleftarrow{h}_1^q; \mathbf{W}^2)\end{aligned}\tag{3.6}$$

Maxpooling-Matching: Figure 3.3(b) illustrates this strategy where each time-step of the premise is matched against each time-step of the hypothesis; then a maxpooling operation is applied to retain only maximum value of each dimension.

$$\begin{aligned}\overrightarrow{m}_i^{max} &= \max_{j \in 1 \dots N} f_m(\vec{h}_i^p, \vec{h}_j^q; \mathbf{W}^3) \\ \overleftarrow{m}_i^{max} &= \max_{j \in 1 \dots N} f_m(\overleftarrow{h}_i^p, \overleftarrow{h}_j^q; \mathbf{W}^4)\end{aligned}\tag{3.7}$$

Attentive-Matching: Figure 3.3 (c) gives the diagram of this matching strategy. This strategy works in similar fashion to the normal attention mechanism (Bahdanau et al., 2014; Luong et al., 2015). We first calculate the weights for each time-step

of hypothesis Q by calculating the cosine similarities between each forward (or backward) time-step \vec{h}_i^p (or \overleftarrow{h}_i^p) and every forward (or backward) time-step of the other sentence \vec{h}_j^q (or \overleftarrow{h}_j^q):

$$\begin{aligned}\overrightarrow{\alpha}_{ij} &= \text{cosine}(\vec{h}_i^p, \vec{h}_j^q) & j = 1, \dots, N \\ \overleftarrow{\alpha}_{ij} &= \text{cosine}(\overleftarrow{h}_i^p, \overleftarrow{h}_j^q) & j = 1, \dots, N\end{aligned}\quad (3.8)$$

After getting the weights, we calculate an attentive vector for the hypothesis Q :

$$\begin{aligned}\vec{h}_i^{\text{mean}} &= \frac{\sum_{j=1}^N \overrightarrow{\alpha}_{ij} \cdot \vec{h}_j^q}{\sum_{j=1}^N \overrightarrow{\alpha}_{ij}} \\ \overleftarrow{h}_i^{\text{mean}} &= \frac{\sum_{j=1}^N \overleftarrow{\alpha}_{ij} \cdot \overleftarrow{h}_j^q}{\sum_{j=1}^N \overleftarrow{\alpha}_{ij}}\end{aligned}\quad (3.9)$$

Finally we match each forward (or backward) time-step with its attentive vector:

$$\begin{aligned}\overrightarrow{m}_i^{\text{att}} &= f_m(\vec{h}_i^p, \vec{h}_i^{\text{mean}}; \mathbf{W}^5) \\ \overleftarrow{m}_i^{\text{att}} &= f_m(\overleftarrow{h}_i^p, \overleftarrow{h}_i^{\text{mean}}; \mathbf{W}^6)\end{aligned}\quad (3.10)$$

Max-Attentive-Matching: This strategy is shown given in Figure 3.3 (d). This is similar to the Attentive-Matching strategy except instead of taking the weighted sum, we take only the time-step having highest weight.

3.2 Visual Bilateral Multi-perspective Matching

In light of Wang et al.'s findings, we propose Visual Bilateral Multi-Perspective Matching framework (V-BiMPM) to solve our proposed task, namely multimodal NLI. Figure 3.4 shows the overview architecture of the framework. As has been noted, we keep the BiMPM architecture and factor in the image. For this purpose, we define a multimodal matching operation to match the image with the text. We represent image as sequence of image features making it possible to match image and text in the same manner as textual matching.

3.2.1 Framework overview

Figure 3.4 show the overview of the framework. Specifically, given an image I , a premise (P), a hypothesis (Q), the framework estimates the probability distribution $P(y|I, P, Q)$. To this end, the text and image are forwarded through the same five layers as in BiMPM framework. The difference is in the *embedding layer* and the *matching layer* where we handle the image.

Embedding layer. Inspired from work in image captioning (Karpathy and Fei-Fei, 2015; Xu et al., 2015a; Fang et al., 2015), we represent image as a set of feature vectors. We utilize the VGG16 model from Simonyan and Zisserman (2014) and

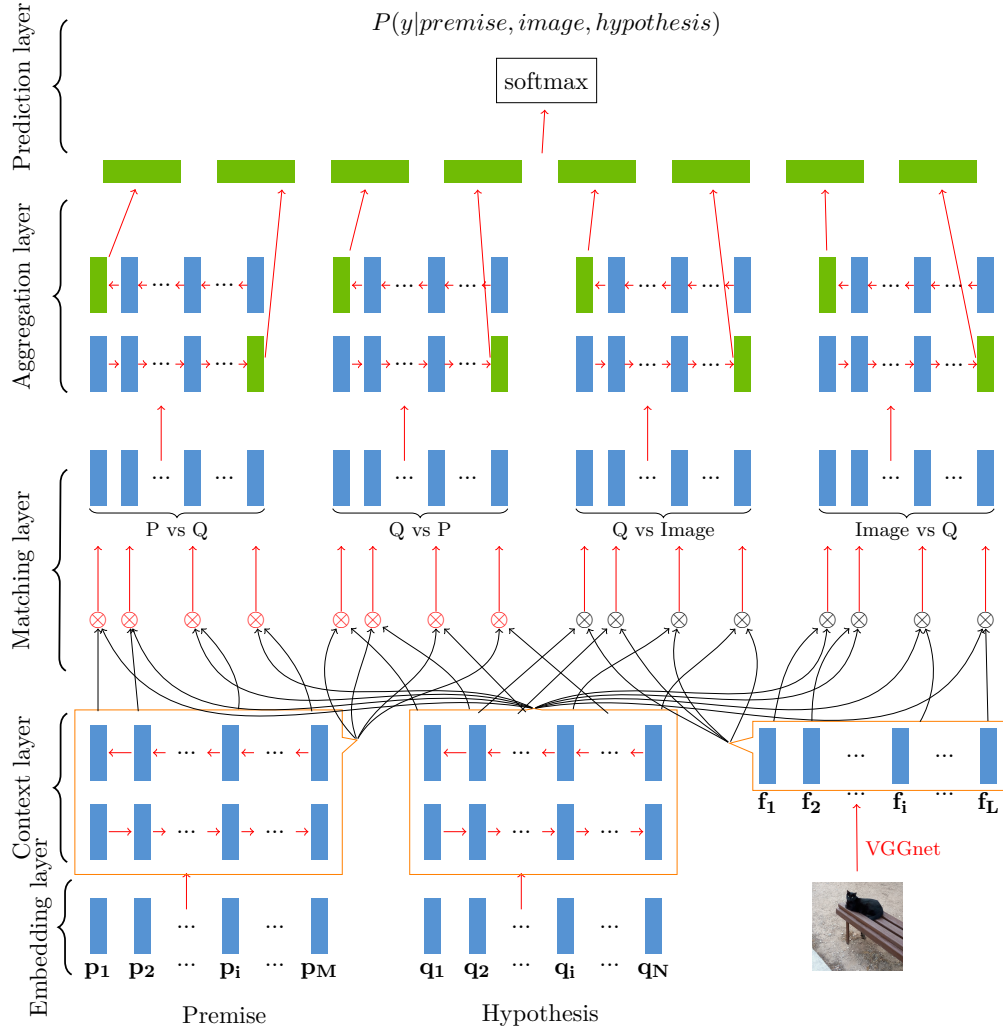


FIGURE 3.4: Architecture of Visual-Bilateral Multi-perspective Matching framework. Due to space limitation, the diagram only shows the hypothesis matched with the image.

extract features before the fully-connected layer which results in a set of feature vectors. This allows the framework to focus on particular parts of the image in the matching process.

$$\text{Image } I : \quad \{f_1, \dots, f_L\}, \quad f_i \in \mathbb{R}^e \quad (3.11)$$

Matching layer. Here, we perform the textual and multimodal matching operations. Before this layer, text is turned into sequences of contextual vectors and image is turned into a set of image features. Initially, the premise is matched against the hypothesis and vice versa as defined in the BiMPM framework. In the same way, the image, the hypothesis, the premise are in turn matched with each other. In Figure 3.4, the textual and multimodal matching operations are denoted by \otimes and \otimes respectively. We use all the matching strategies described in BiMPM framework.

3.2.2 Multi-modal Multi-perspective Operation

This is the crucial part in the model where we match the textual vector to visual vectors and vice versa. Concretely, we need to calculate cosine similarities between textual vectors and visual vector. They not only have different dimensions but also belong to different spaces. In this thesis, we project these vectors into a mutual space using affine transformation.

$$v_i = \mathbf{W}_t f_i + b_t; \quad f_i \in \mathbb{R}^e; \mathbf{W}_t \in \mathbb{R}^{e \times d}; b_t, v_i \in \mathbb{R}^d \quad (3.12)$$

Similar to the textual matching operation, output of this operation is a l -dimension vector. l is the number of perspectives.

$$m = f_m(v_t, v_i; \mathbf{W}; \mathbf{U}) \quad (3.13)$$

where v_t and v_i are textual and visual vectors. $\mathbf{W} \in \mathbb{R}^{l \times d}$ and $\mathbf{U} \in \mathbb{R}^{l \times d}$ are weight matrices for textual and visual spaces. m is a l -dimensional matching result vector $m = [m_1, \dots, m_l]$. Each element in m is calculated by

$$m_k = \text{cosine}(W_k \circ v_t, U_k \circ v_i) \quad (3.14)$$

3.3 Conclusion

In this chapter, we have discussed the baseline model and our proposed model. We choose BiMPM framework (Wang et al., 2017) as the baseline model. This framework implements a strong matching operation together with elegant attention strategies; however it only works with text. Our proposed model is an extension of BiMPM where we integrate visual information into the framework. This integration requires a cross-modality matching between textual and visual vectors. In this work, we use affine transformation to covert both modalities into a common space before matching them.

In the next chapter, we present our data collection process and performance of discussed models on this dataset.

Chapter 4

Experiment results

In this chapter, we present the tests carried out to examine our proposed model. To do this, we first form a multimodal dataset from SNLI and Flickr30k. Then we run the proposed multimodal model and the textual models on the same dataset with the same hyper-parameters. The experiment results and analysis show the effectiveness of the proposed model.

4.1 Dataset

The Stanford Natural Language Inference (SNLI) Corpus was created by Bowman et al. (2015) for the purpose of NLI benchmarking. Modern methods on NLI, or NLP in general are data-intensive and not domain-restricted. However, the current available datasets for NLI are relatively small and are not suitable for those methods. For example, the most recent data set, the Sentences Involving Compositional Knowledge (SICK) from SemEval 2014 task (Marelli et al., 2014) contains 10,000 sentence pairs which are augmented from 1500 original pairs. Meanwhile, SNLI contains 570,152 sentence pairs generated by 2500 annotators (Bowman et al., 2015).

SNLI is manually created by employing crowdsourcing. The premises are image captions borrowed from Flickr30k (Young et al., 2014). This Flickr30k dataset contains 160k captions and was also created by crowdsourcing. It is important to notice that annotators are informed that these premise are photo captions. Given the premise, workers from Amazon Mechanical Turk are asked to provide *true*, *neutral* and *false* descriptions to the photo. The workers are encouraged to produce novel sentences without sentence length limitation.

SNLI solved the problem of indeterminacies of event and entity coreference. This problem reduces the agreement rate on determining the correct semantic labels. For example, consider the two sentences *cheerleaders are on the field cheering* and *some people are cheering on a field*. If we assume "*cheerleaders*" and "*some people*" referring to the same people then the pair will be labeled as entailment, otherwise it could be neutral. Likewise, we also have to choose whether we should make an assumption on event coreference. SNLI solved this issue by grounding the premise and hypothesis sentences in a specific scene. As a result, 58% of cases show a unanimous consensus from all five annotators and 98% of cases see a three annotator consensus (Bowman et al., 2015).

To create a multimodal NLI dataset, we match each sentence pair to the corresponding image in the Flickr30k dataset. A small subset of the SNLI dataset cannot be match with any image, therefore our final data is slightly smaller than the SNLI. As shown

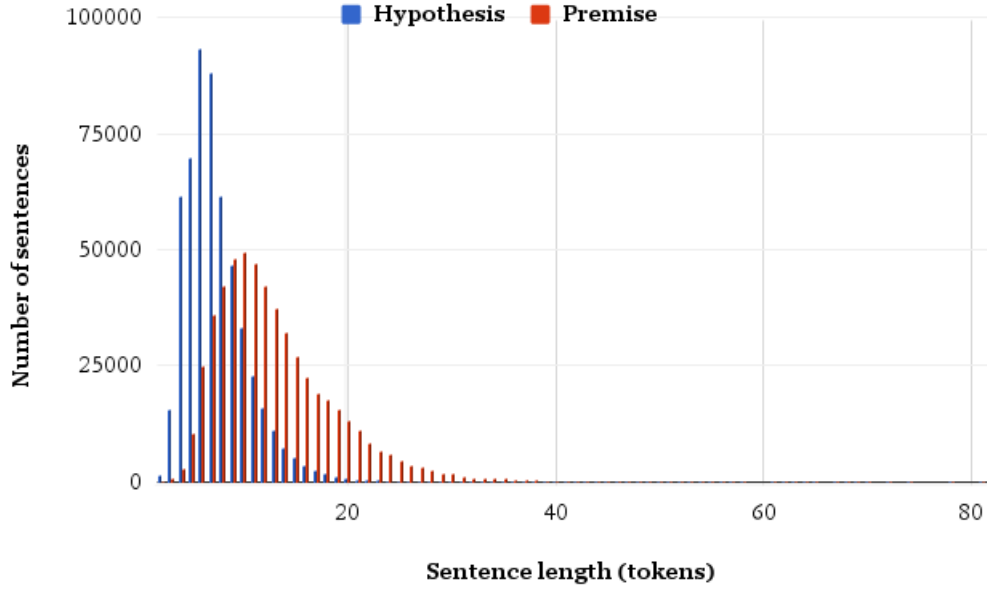


FIGURE 4.1: Distribution of sentence lengths for premise and hypothesis sentences

in table 4.1, the data is distributed evenly among all the classes. Meanwhile, figure 4.1 shows that premise sentences vary considerably and hypothesis sentences tend to be short. Similar figures and statistics for SNLI are reported in Bowman et al. (2015). Finally, Table 4.2 shows some examples taken from our final training set.

Adding the image gives us more information but occasionally could make the gold standard label of the neutral instances questionable at the same time. Consider the last example in Table 4.2, *A big dog catches a ball on his nose* and *A big dog is sitting down while trying to catch a ball*. Without the image, one would classify this sentence pair as neutral but it is clearly contradiction if we take the corresponding image into account. This is because information about the posture of the dog is present in the image but not in the premise. This is also an example of how images can convey more than what is in the text. However, as we analyze a small sample of the testing set (due to the large volume of this dataset), this only affects less than 1% of the data.

Dataset	Entailment	Neutral	Contradiction	Total
Train	182,167	181,515	181,938	545,620
Test	3,368	3,219	3,237	9824
Dev	3,329	3,235	3,278	9842

TABLE 4.1: Data distribution on the obtained dataset

4.2 Hyperparameters

All models are implemented in Tensorflow v1.1¹. Wang et al., 2017 provide the implementation for BiMPM on the earlier version of Tensorflow². Following this

¹<https://www.tensorflow.org/>

²<https://github.com/zhiguowang/BiMPM>

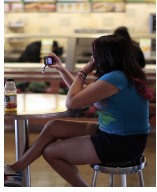




Flickr30k id	Premise	Label	Hypothesis
	A woman sitting at a table , taking a picture .	entailment	The woman is sitting down
	People sit and relax next to a pool in a plaza .	neutral	The pool is in a hotel , and these are guests
	A man parasails in the choppy water .	contradiction	The man parasailed in the calm water
	Busy ChinaTown street corner where people are walking past an open front store .	entailment	There are people in ChinaTown
	A big dog catches a ball on his nose	neutral	A big dog is sitting down while trying to catch a ball.

TABLE 4.2: Example pairs taken from the training set

implementation, we implement our model on newer version of Tensorflow and also make it publicly available³. In addition, Table 4.3 explains some parameters and their values. To focus on the impact of visual information, the values of these parameters are kept unchanged across models.

We use Adam optimizer (Kingma and Ba, 2014) to train all the models. This is a method for stochastic optimization which decrease the cost function by updating the model parameters on a subset of the training instances. In this thesis, we choose the cost function to be the cross entropy error function and regularization L_2 . Each subset or batch, in our experiments, contains 180 instances with premises having similar lengths. Each model is trained in 10 epochs, the best-performing model on development set are picked. Dropout rate and learning rate are fixed to 0.1 and 0.001 respectively.

4.3 Results

Table 4.4 shows the average accuracies for the different models over three train and test iteration. The first model, denoted by Hypo \otimes Prem, is the textual BiMPM model. We obtain an accuracy of 86.41% for this model which is slightly lower than one reported by Wang et al. (2017). This can be explained by the fact that we have to restrict the vocabulary size due to hardware limitation. The best-performing model

³<https://github.com/hoavt-54/nli-images>

Parameter	Explanation	Value
max_char_per_word	Character limit for character embedding	10
max_sent_length	Sentence length limit	100
lambda_l2	Regularization L2 value	0.0
fix_word_vec	Whether to train word embedding with the model	false
char_lstm_dim	Dimension of LSTM when encoding character embedding	100
char_emb_dim	Dimension of character embedding in the final word embedding	20
context_lstm_dim	Dimension of LSTM in the Context layer	100
aggregation_lstm_dim	Dimension of LSTM in the Aggregation layer	300
vocab_size	Vocabulary size	300,000
learning_rate	Learning rate	0.001
dropout_rate	Dropout rate	0.1
max_epochs	Number of epochs	10
MP_dim	Number of perspectives in the matching operation	10
batch_size	Number of instances in each batch	180

TABLE 4.3: Hyperparameters for experiments

which matches the hypothesis with the premise and the image yields an accuracy of **87%** approximately. Thus, the inclusion of the image in the BiMPM architecture yields more than a half percentage increase in accuracy.

System	Accuracy	
	Train	Test
Hypo \otimes Prem	92.11	86.41
Hypo \otimes Image	83.32	73.50
Prem \otimes Image + Hypo \otimes Image	87.43	82.19
Hypo \otimes Prem + Hypo \otimes Image	91.15	86.99
Hypo \otimes Prem + Hypo \otimes Image + Prem \otimes Image	91.23	86.81

TABLE 4.4: Average accuracies of different models after 3 times running. *Hypo* stands for *hypothesis*, *prem* stands for *premise*

We performed an ablation study to highlight different components in the models. First, we removed the textual matching operations between the premise and the hypothesis. Then, we continue to remove the matching operation between the premise and the image. Interestingly, the model that does not include textual matching still performs quite well (73.5 and 82.19). This proves that visual information plays an important role in these models. However, the low accuracies in the training set suggest that these models are underfitting, which is reasonable due to its simplicity.

4.4 Analysis

We further break down the results by analyzing outputs from the textual and multimodal systems. Table 4.5 shows performance of these two models on each class over there training and evaluation iteration. As clearly seen, the multimodal outperforms the textual model significantly on the contradiction and entailment class. This shows that visual information is helpful to remedy the ambiguities of natural language.

On the neutral class, however, the multimodal model only performs slightly better. One of the possible reasons is what we discussed in Section 4.1 namely that adding visual information occasionally can make gold standard label questionable.

Label	Prediction	Occurrence	
		BiMPM	V-BiMPM
contradiction	contradiction	8376	8500
contradiction	entailment	421	392
contradiction	neutral	914	819
entailment	contradiction	154	205
entailment	entailment	9097	9132
entailment	neutral	853	767
neutral	contradiction	587	614
neutral	entailment	1075	1036
neutral	neutral	7995	8007

TABLE 4.5: Detailed performance of textual and multimodal system on each class.

4.5 Summary

In this section, we have seen that visual information is useful for NLI task. The best-performing multimodal model gains 0.6% over the textual model. This increase is markedly meaningful if we consider the fact that performances of the textual model is already approaching the human performance (Gong et al., 2017). It is interesting that even matching only the image with the hypothesis yields a non-trivial result. This suggests that information from the image can be effectively exploited for the NLI task.

Chapter 5

Conclusions and Future work

5.1 Conclusion

In this thesis, we proposed the *Multimodal Natural Language Inference* task which is an extension of Natural Language Inference. Given an image and its description as the premise, the task is to determine whether a hypothesis contradicts, entails or is neutral with regards to the image and its description. The task is challenging not only due to the natural complexing of human languages, but also because it is not obvious how to incorporate visual information into a textual model.

The dataset for this task was based upon the SNLI (Stanford Natural Language Inference) dataset. SNLI was manually created by asking annotators to come up with hypothesis which either entails, contradict or is neutral with respect to an image description. For our purpose, we mapped each sentence pair of SNLI to the original image dataset Flickr30k. The result dataset, made publicly available, has more than 565k instances.

Experiments show the usefulness of visual information in solving natural language inference. First, we start with the NLI state-of-the-art framework BiMPM (Wang et al., 2017). We extend this model by incorporating visual information and adding a multimodal matching operation. Experiments on our newly formed dataset show that our multimodal framework outperforms the original BiMPM.

5.2 Contribution

We would like to highlight some of the main contributions of this thesis. These contributions include:

- We carried out one of the first experiments on including visual information for NLI.
- We created a multimodal dataset for NLI based on SNLI.
- We achieve an advance over the current, best performing system in the NLI task.

5.3 Future work

Possible future work includes extending the current proposed model and applying BiMPM to other multimodal tasks. In this thesis, the two modality text and visual are matched using affine transformation and cosine similarity. Despite the relatively simple similarity metric which directly compares vector in different spaces, experiments shows an encouraging improvement. Having this result at hand, we would like to explore other methods to deal with multimodality, for example those proposed by Socher et al. (2014) and Karpathy and Fei-Fei (2015) in developing multimodal spaces.

Applying BiMPM framework to other multimodal tasks is a promising research direction. In fact, this framework achieve 73.5% by matching only the hypothesis to the image. Therefore, a BiMPM-like model can be applied to other tasks similar to Multimodal NLI like, Visual Question Answering or Image captioning.

Bibliography

- Akhmatova, Elena (2005). “Textual entailment resolution via atomic propositions”. In: *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*. Vol. 150.
- Andrews, Mark, Gabriella Vigliocco, and David Vinson (2009). “Integrating experiential and distributional data to learn semantic representations.” In: *Psychological review* 116.3, p. 463.
- Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). “Neural machine translation by jointly learning to align and translate”. In: *preprint arXiv:1409.0473*.
- Barron, Andrew R (1993). “Universal approximation bounds for superpositions of a sigmoidal function”. In: *IEEE Transactions on Information theory* 39.3, pp. 930–945.
- Bayer, Samuel et al. (2005). “MITRE’s Submissions to the EU Pascal RTE Challenge”. In: *Proceedings of the Pattern Analysis, Statistical Modelling, and Computational Learning (PASCAL) Challenges Workshop on Recognising Textual Entailment*.
- Bengio, Yoshua et al. (2007). “Greedy layer-wise training of deep networks”. In: *Advances in neural information processing systems*, pp. 153–160.
- Bernardi, Raffaella et al. (2016). “Automatic Description Generation from Images: A Survey of Models, Datasets, and Evaluation Measures.” In: *J. Artif. Intell. Res.(JAIR)* 55, pp. 409–442.
- Bos, Johan and Katja Markert (2006). “When logical inference helps determining textual entailment (and when it doesn’t)”. In: *Proceedings of the Second PASCAL RTE Challenge*, p. 26.
- Bowman, Samuel R et al. (2015). “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.
- Bowman, Samuel R et al. (2016). “A fast unified model for parsing and sentence understanding”. In: *arXiv preprint arXiv:1603.06021*.
- Chen, Qian et al. (2017). “Recurrent neural network-based sentence encoder with gated attention for natural language inference”. In: *arXiv preprint arXiv:1708.01353*.
- Dagan, Ido, Oren Glickman, and Bernardo Magnini (2006). “The PASCAL recognising textual entailment challenge”. In: *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising textual entailment*. Springer, pp. 177–190.
- Deng, Li, Dong Yu, et al. (2014). “Deep learning: methods and applications”. In: *Foundations and Trends® in Signal Processing* 7.3–4, pp. 197–387.
- Fang, Hao et al. (2015). “From captions to visual concepts and back”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1473–1482.
- Geffet, Maayan and Ido Dagan (2005). “The distributional inclusion hypotheses and lexical entailment”. In: *Proceedings of the 43rd Annual Meeting on Association for*

- Computational Linguistics*. Association for Computational Linguistics, pp. 107–114.
- Glickman, Oren and Ido Dagan (2005). “Web based probabilistic textual entailment”. In: *In Proceedings of the 1st Pascal Challenge Workshop*.
- Goldberg, Yoav (2016). “A Primer on Neural Network Models for Natural Language Processing.” In: *J. Artif. Intell. Res.(JAIR)* 57, pp. 345–420.
- Gong, Yichen, Heng Luo, and Jian Zhang (2017). “Natural language inference over interaction space”. In: *arXiv preprint arXiv:1709.04348*.
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press.
- Graves, Alex and Jürgen Schmidhuber (2005). “Framewise phoneme classification with bidirectional LSTM and other neural network architectures”. In: *Neural Networks* 18.5, pp. 602–610.
- Harris, Zellig S (1954). “Distributional structure”. In: *Word* 10.2-3, pp. 146–162.
- He, Kaiming et al. (2016). “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.
- Hinton, Geoffrey E, Simon Osindero, and Yee-Whye Teh (2006). “A fast learning algorithm for deep belief nets”. In: *Neural computation* 18.7, pp. 1527–1554.
- Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780.
- Hornik, Kurt (1991). “Approximation capabilities of multilayer feedforward networks”. In: *Neural networks* 4.2, pp. 251–257.
- Jijkoun, Valentin, M Rijke, et al. (2005). “Recognizing textual entailment using lexical similarity”. In: *Recognizing Textual Entailment*, p. 73.
- Karpathy, Andrej and Li Fei-Fei (2015). “Deep visual-semantic alignments for generating image descriptions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3128–3137.
- Kingma, Diederik and Jimmy Ba (2014). “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980*.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*, pp. 1097–1105.
- LeCun, Yann et al. (1989). “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4, pp. 541–551.
- Lin, Zhouhan et al. (2017). “A structured self-attentive sentence embedding”. In: *arXiv preprint arXiv:1703.03130*.
- Ling, Wang et al. (2015). “Finding function in form: Compositional character models for open vocabulary word representation”. In: *arXiv preprint arXiv:1508.02096*.
- Luong, Minh-Thang, Hieu Pham, and Christopher D Manning (2015). “Effective approaches to attention-based neural machine translation”. In: *arXiv:1508.04025*.
- Lynch, Corey, Kamelia Aryafar, and Josh Attenberg (2016). “Images don’t lie: Transferring deep visual semantic features to large-scale multimodal learning to rank”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 541–548.
- MacCartney, Bill (2009). *Natural language inference*. Stanford University.
- Marelli, Marco et al. (2014). “A SICK cure for the evaluation of compositional distributional semantic models.” In: *LREC*, pp. 216–223.
- McCulloch, Warren S and Walter Pitts (1943). “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133.

- Mikolov, Tomáš (2012). “Statistical language models based on neural networks”. In: *Presentation at Google, Mountain View, 2nd April*.
- Mikolov, Tomas et al. (2010). “Recurrent neural network based language model.” In: *Interspeech*. Vol. 2, p. 3.
- Mou, Lili et al. (2016). “Natural language inference by tree-based convolution and heuristic matching”. In: *The 54th Annual Meeting of the Association for Computational Linguistics*, p. 130.
- Nangia, Nikita et al. (2017). *The RepEval 2017 Shared Task: Multi-Genre Natural Language Inference with Sentence Representations*. Copenhagen, Denmark.
- Pasunuru, Ramakanth and Mohit Bansal (2017). “Multi-Task Video Captioning with Video and Entailment Generation”. In: *arXiv preprint arXiv:1704.07489*.
- Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.
- Rosenblatt, Frank (1958). “The perceptron: A probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6, p. 386.
- Roy, Deb (2005). “Semiotic schemas: A framework for grounding language in action and perception”. In: *Artificial Intelligence* 167.1-2, pp. 170–205.
- Simonyan, Karen and Andrew Zisserman (2014). “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556*.
- Socher, Richard et al. (2014). “Grounded compositional semantics for finding and describing images with sentences”. In: *Transactions of the Association for Computational Linguistics* 2, pp. 207–218.
- Specia, Lucia et al. (2016). “A Shared Task on Multimodal Machine Translation and Crosslingual Image Description.” In: *WMT*, pp. 543–553.
- Sundermeyer, Martin, Ralf Schlüter, and Hermann Ney (2012). “LSTM neural networks for language modeling”. In: *Thirteenth Annual Conference of the International Speech Communication Association*.
- Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems*, pp. 3104–3112.
- Wang, Zhiguo, Wael Hamza, and Radu Florian (2017). “Bilateral Multi-Perspective Matching for Natural Language Sentences”. In: *arXiv preprint arXiv:1702.03814*.
- Xu, Kelvin et al. (2015a). “Show, attend and tell: Neural image caption generation with visual attention”. In: *International Conference on Machine Learning*, pp. 2048–2057.
- Xu, Wenduan, Michael Auli, and Stephen Clark (2015b). “CCG Supertagging with a Recurrent Neural Network.” In: *ACL (2)*, pp. 250–255.
- Young, Peter et al. (2014). “From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions”. In: *Transactions of the Association for Computational Linguistics* 2, pp. 67–78.
- Zeiler, Matthew D and Rob Fergus (2014). “Visualizing and understanding convolutional networks”. In: *European conference on computer vision*. Springer, pp. 818–833.

List of Figures

1.1	A "sentence encoder" architecture.	5
2.1	An example of a 2-layer feed-forward network where inputs, hidden units, output are denoted by nodes and weights are represented by arrows	9
2.2	Calculate error δ_j at hidden unit j by backpropagate (dash arrows) errors from all the nodes that j sends information	11
2.3	Caption for LOF	12
2.4	An example of a convolutional layer. The input image having size 32x32 with 3 color channels. After applying a set of filter, each filter return a 2D feature map, we obtain a 3D output tensor.	13
2.5	A recurrent neural network unrolled through time.	14
2.6	Example of Acceptor RNNs	16
2.7	An example of RNN-based language model working as a transducer	16
2.8	An encoder-decoder translating sequence " $x_0x_1x_2...x_t$ " to sequence " $y_0y_1y_2...y_t$ "	16
2.9	An example of image captioning system from Karpathy and Fei-Fei (2015)	17
3.1	Overview of BiMPM architecture.	20
3.2	Word embedding components in BiMPM	20
3.3	Diagram for 4 matching strategies	22
3.4	Architecture of Visual-Bilateral Multi-perspective Matching framework. Due to space limitation, the diagram only shows the hypothesis matched with the image.	24
4.1	Distribution of sentence lengths for premise and hypothesis sentences	28

List of Tables

4.1	Data distribution on the obtained dataset	28
4.2	Example pairs taken from the training set	29
4.3	Hyperparameters for experiments	30
4.4	Average accuracies of different models after 3 times running. <i>Hypo</i> stands for <i>hypothesis</i> , <i>prem</i> stands for <i>premise</i>	30
4.5	Detailed performance of textual and multimodal system on each class.	31

List of Abbreviations

BiMPM	B ilateral M ulti- P erspective M atching
BiLSTM	B idirectional L ong- S hort T erm M emory
CovNets	C ovolutional N eural N etworks
CV	C omputer V ision
NLI	N atural L anguage I nterpretation
SNLI	S tanford N atural L anguage I nterpretation
RNNs	R ecurrent N eural N etworks