ON LEXICAL LEVEL MATCHING

BY

SHAOSHI LING

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Adviser:

Professor Dan Roth

ABSTRACT

In many natural language understanding applications, text processing requires comparing lexical units: words, phrases, name entities and sentences. A significant amount of research has taken place in studying evaluating similarity metrics between those units. In this thesis, we summarize some research work in computing lexical similarity. We describe a new approach to compute similarity between two spans of text, using multiple semantic-units level comparison measures to compute sentence-level similarity scores.

To my parents, for their love and support.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Prof. Dan Roth, for all the supports in the last three years. Dan Roth has been a perfect advisor. I really enjoy solving challenging problems with him and I always admire his insightful ideas. Dan has steered my research and yet given me the intellectual freedom to pursue my interests. He demonstrates what is a remarkable scholar and a passionate educator.

I would like to thank everyone I encountered in the last five years. They shaped me and made my life in Champaign-Urbana more colorful. I especially acknowledge the members in CogComp group. Over the years, I have interacted and worked closely with many people belonging to the group. No part of my research would have been possible without help from them.

Finally, and most importantly, I would like to thank my parents, my grandparents, my girlfriend, for their unconditional sacrifice and support. I will not be able to go this far without you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

LLM	Lexical Level Matching
NER	Name Entity Recognition
NLP	Natural Language Processing
NESim	Name Entity Similarity

CHAPTER 1

INTRODUCTION

Many natural language understanding applications need to compute the similarity between sentences or between pairs of other short snippets as one of their fundamental operation. For instance, in document summarization tasks [29] [2], there is a need to identify and remove duplicate sentences to maximize the diversity of information in a summary. Similarly, computing semantic similarity between sentences is essential in evaluating automated paraphrasing techniques [6]. In textual entailment tasks [30], there is a need to decide if two sentences refer to the same concepts, even though they may be using different words to express them. Large news aggregators need to identify similar news reports so that they can cluster or remove duplicate news articles received from various news agencies.

1.1 Lexical Similarity

One approach to sentence similarity is matching their lexical tokens between two sentences. So Lexical similarity plays an important role in our method to text similarity. Lexical similarity is a measure of the degree to which the lexical tokens of two given languages are similar. The lexical tokens include words, phrases, and name-entity. One very common approach to this problem is to train and obtain a representation of those tokens and use the representation to compute the degree of similarity.

1.2 Challenges and Motivation

With the trend of deep learning, the NLP community shows a strong interest in the study of sentence representations [5][14]. Their goal is to use the neural language model to represent a span of text as a fixed-length feature vector. Their method achieves the the-state-of-art performance on various tasks including sentence level similarity. However, to convert a sentence into a feature vector is a very complicated process including a large amount of training and time-consuming encoding process. Our motivation is to develop a light and simple approach to compute sentence similarity.

Lexical level matching is one of traditional word alignment method [8]. With years of development, this approach is well-studied, very robust and most importantly is in light-weight.

One problem is that in the current lexical level matching algorithm, tokens representation methods are simply based on the single word. They regard each word independently and employ single word as a basic unit. However, from a cognitive point of view, it can be argued that the basic units that the human cognitive system uses include not only single words but also multiple-word phrases and name entities. One limitation of the current implementation is the comparing representation of non-compositional phrases. Particularly convincing examples for such units are phrasal verbs in English, which often have a non-compositional meaning. For example, it is more plausible that we recognize “keep up”, “keep on” and “keep from” as relevant basic linguistic units in these contexts and that the human cognitive systems represent them as units. The other limitation is that it’s hard to handle name entity with typing. For example, the word “Washington” can both refer to location “Washington State” and the person “George Washington”. Using unigram representation methods fail to be discriminative for those token units.

The goal of our work is to improve the performance of current lexical level matching (LLM) implementation. We extend lexical similarity from word-based to the semantic-unit based comparison. Our idea is to treat word, phrase, name entity with typing as a basic unit to handle limitation of unigram-based representation in lexical level matching.

1.3 Overview of the Work

To achieve this goal, we first use data mining approach to extract semantic units in a large corpus and then train embedding representation for them. Our semantic relatedness experiments show that employing semantic-unit based representations in LLM outperforms the LLM with only traditional word-based representation.

In chapter 2, we discuss the background information including LLM, name entity recognition approach and phrases mining methods. In chapter 3, we discuss the approach to text similarity by LLM using different word representation. In chapter 4, we obtain semantic-unit representation and explain our approach to tokenize the text into word, phrases and name-entity. In chapter 5, we propose the new LLM which uses phrases and name-entity comparison to compute sentence similarity. In chapter 6, we conduct experiments to show the effectiveness of semantic unit matching against normal word level matching. And in the last chapter, we made the conclusion and list the future work.

CHAPTER 2

BACKGROUND

We started by discussing some very related topic and the existing algorithms which will be the foundation of building our new LLM similarity metrics.

2.1 Word Similarity Metrics

Word similarity, also referred as word semantic similarity, is a metric to evaluate the distance between them based on the likeness of their meaning or semantic content as opposed to similarity which can be estimated regarding their syntactical representation (e.g. their string format).

Wordnet similarity is one of the commonly used approaches. It implements measures of similarity and relatedness that are all in some way based on the structure and content of WordNet which is an on-line lexical reference system [24].

And there are a large number of other word semantic similarity measures, using approaches that are either knowledge based or corpus-based [18]. One knowledge-based word representation we will use is Explicit Semantic Analysis, a method that represents the meaning of texts in a high-dimensional space of concepts derived from Wikipedia.[11].

In recent year, the use of dense distributional lexical representations, known as word embeddings, supports better performance on a range of NLP tasks [3][15] [19]. Thus, word embeddings have been commonly used in the last few years for lexical similarity tasks and as features in multiple, syntactic and semantic, NLP applications.

We use those metrics in LLM as word comparator and we will discuss each of them in very detail in Chapter 3.

2.2 Lexical Level Matching

The similarity between two spans of text is computed based on the individual term-similarity as follows: First, both sentences are tokenized to find all semantic units, viz. named entities, phrasal verbs, multi-word expressions, and words. Then, the similarity metrics are applied based on the type of semantic units to match the units from one sentence to the most similar unit from the other sentence. At the end of this step, all semantic units map to their best counterparts from the other sentence. Finally, the sentence-level similarity score is computed as the sum of the similarity scores of the matching pairs, normalized by the number of units matched. We refer to this measure as the Lexical Level Matching (LLM) score. For two sentences s_1 and s_2, such that $|s_1| \geq |s_2|$,

$$LLM(s_1, s_2) = \frac{\sum_{v \in s_1} max_{u \in s_1} sim(u, v)}{|s_2|} \tag{2.1}$$

where sim(u, v) is the corresponding similarity metrics defined over semantic units u and v [8].

2.3 Phrases Mining

This section introduces phrases mining technique which we covered in phrase representation step in the new approach.

Comparing with unigrams (single word), a natural, meaningful, unambiguous semantic unit (phrase) is more effective to manipulate unstructured text data. We are supposed to generate high quality phrases, which should have high popularity, concordance, completeness and meaningful, to help us manipulate unstructured text data. In overall, we can transform unigrams to the semantic unit (phrase) based text processing.

In natural language processing field, the community has conducted extensive studies typically referred to as automatic term recognition [10][33], for the computational task of extracting terms (such as technical phrases). Supervised noun phrase chunking techniques [26] exploit such tagged documents to automatically learn rules for identifying noun phrase boundaries. The dependency on these various kinds of linguistic analyzers, domain-dependent language rules, and expensive human labeling, makes it challenging to extend these approaches to emerging, big, and unrestricted corpora, which may include many different domains, topics, and languages [28].

In data mining field, there are many data-driven and unsupervised approaches can overcome this limitation. They make use of frequency statistics in the corpus to address both candidate generation and quality estimation [16][7][23]. Which means they no longer need to rely on complex linguistic feature generation, domain-specific rules [16]. The basic idea is based on frequent pattern mining. If the probabilities of co-occurrence of certain words are high, which means they are highly frequent, we can determine that those words patterns have high probabilities to be phrases. However, even the probabilities of co-occurrence of certain words are pretty high, we can not ensure those words patterns are phrases because they may not discriminative and informative, eg. this paper or they may lose completeness such as vector machine vs support vector machine. The general principle of approaches is exploiting information redundancy and data-driven criteria to determine phrase boundaries and salience.

2.4 Name Entity Recognition

In this section, we introduce name entity recognition technique we used to process input sentences in LLM.

A named entity is a sequence of words that designate some real-world entity, e.g. "California", "Steve Jobs" and "Apple Inc." The task of named entity recognition is to identify named entities from the free-form text and to classify them into a set of predefined types such as a person, organization and location[1]. In another word, Named entity recognition (NER) is the problem of locating and categorizing important nouns and proper nouns in a text[22]. For example, give a sentence

The best **BBQ** I've tasted in **Phoenix**! I had the **pulled pork sandwich** with **coleslaw** and **baked beans** for lunch.

The highlighted named entities hold most of the important information in the sentence, and are phrases we care the most in various natural language processing applications.

Named entity recognition is probably the most fundamental task in information extraction [1]. To extract complex structures such as relations and location-based events, an accurate named entity recognition is an important preprocessing step. Other than that, named entity recognition is also widely used in Question Answering [13] and Machine Translation. In question answering, the candidate answer strings are always named entities, which need to be extracted first using NER. In machine translation, for example, the extracted entity *Support Vector Machine* avoid the machine to translate the phrase word by word, which is not exactly the correct meaning.

In our LLM, we use CogCom NLP NER package to recognize name-entity in processing sentences [27].

CHAPTER 3

LEXICAL SIMILARITY USING WORD COMPARATOR

In LLM with word comparator, sentences are tokenized to find all word units. Then, we apply word similarity metrics to match the word from one sentence to the most similar word from the other sentence.

Like shown in equation:

$$LLM(s_1, s_2) = \frac{\sum_{v \in s_1} max_{u \in s_1} sim(u, v)}{|s_2|} \tag{3.1}$$

where we use word similarity metric as sim(u, v) in the equation [8].

In this chapter, we will talk about multiple word similarity metrics we used in detail.

3.1 LLM With WordNet Hierarchy

One of the most influential works has been in building WordNet [24]. WordNet organizes words into synsets that are further linked to other synsets using hypernymy, meronymy, and other linguistic relations. There have been many similarity metrics proposed using the hierarchical structure of WordNet, especially for nouns [24].

We formulate a similarity measure, WNSim, over the WordNet hierarchy to compute similarity between words. For two words w_1 and w_2 in the WordNet hierarchy, WNSim finds the closest common ancestor of the words, sometimes referred to as least common subsumer (lcs). The similarity is then defined based on the distance of the words from the lcs, as follows:

$$WNSim(w_1, w_2) = \begin{cases} \Theta^{l_1+l_2} & \text{if } l_1 + l_2 \leq k \\ \Theta^{k} & \text{if } l_1 + l_2 \leq \alpha * \text{depth of } lcs(w_1, w_2) \\ o & \text{if } otherwise \end{cases} \tag{3.2}$$

This measure captures the key concepts of hierarchical similarity used in other WordNet based similarity measures. It has 3 parameters: , k, and . In the experiments, we empirically set them as = 0.3, k = 3, and = 0.667, after manually searching over various values for these parameters. The words are first converted to the same part-of-speech, by finding the base verb or noun form of the word, if available, before the appropriate WordNet hierarchy is considered. To compute the least common subsumer, we consider the synonymy-antonymy, hypernymy-hyponymy, and meronymy relations. If the path from the lcs to one of the words contains an antonymy relation, we reduce the similarity value by half and negate the score. Hence, under this scheme, synonyms get a score of 1.0 and antonyms get a similarity value of 0.5. Further, we compare the determiners and prepositions separately if two words are determiners or prepositions, they get a similarity score of 0.5. Hence, this similarity measure gives a score in [1, 1] range. (The motivation here is to discount differences between words that tend to have little influence on overall similarity judgments different prepositions, for example, may take on similar meanings based on context).

3.2 LLM With Word Embedding

Word embeddings have been exceptionally successful in many NLP tasks including word similarity tasks.

The term word embedding was originally coined by [3]. The classical neural model consists of a one-hidden layer feed-forward neural network that predicts the next word in a sequence. The model maximizes the training corpus penalized log-likelihood:

$$L = \frac{1}{T} \sum log f(w_t, w_{t-1}, ..., w_{t-n+1}; \Theta) + R(\Theta) \tag{3.3}$$

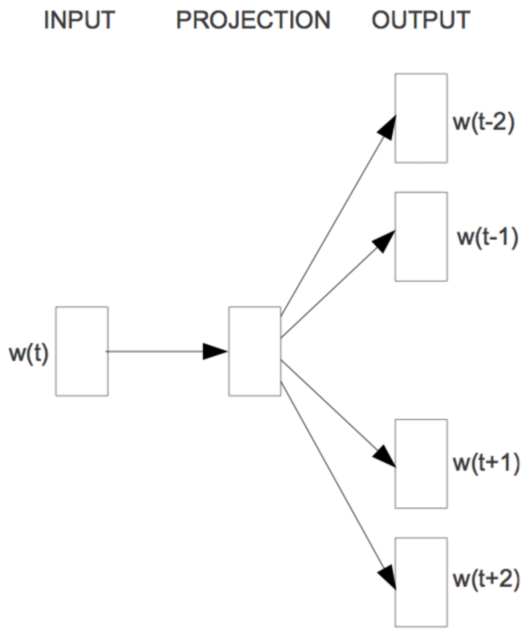
where $R(\Theta))$ is a regularization term. i.e. the probability $p(w_t|w_{t1}, , w_{tn+1})$ as computed by the softmax, where n is the number of previous words fed into the model.

3.2.1 Word2vec

It was Word2vec [19] who really brought word embedding to the forefront through the creation of word2vec, a toolkit enabling the training and use of pre-trained embeddings. Word2Vec is the most popular of the word embedding models. Word2vec recommends two architectures for learning word embeddings: Continuous bag-of-words and Skip-gram. Skip-gram uses the center word to predict the surrounding words as can be seen in Figure 3.1. The skip-gram objective thus sums the log probabilities of the surrounding n words to the left and to the right of the target word wt to produce the following objective:

$$J_\Theta = \frac{1}{T} \sum_{T}^{t=1} \sum_{-n \leq j \leq n, \neq 0} logp(w_{t+j}|w_t) \tag{3.4}$$

Figure 3.1: Skip-gram Model



3.2.2 GloVe

GloVe [25] seeks to make explicit what SGNS does implicitly: Encoding meaning as vector offsets in an embedding space – seemingly only a serendipitous by-product of word2vec – is the specified goal of GloVe.

Specifically, the authors of Glove show that the ratio of the co-occurrence probabilities of two words (rather than their co-occurrence probabilities themselves) is what contains information and aim to encode this information as vector differences. To achieve this, they propose a weighted least squares objective JJ that directly aims to minimize the difference between the dot product of the vectors of two words and the logarithm of their number of co-occurrences:

$$J = \sum_{i,j=1}^{V} f(X_ij)(w_i^T \widetilde{w_j} + b_i + \widetilde{b_j} - logX_{ij})^2 \tag{3.5}$$

where w_i and b_i are the word vector and bias respectively of word i, $w_j$and b_j are the context word vector and bias respectively of word j, X_{ij} the number of times word i occurs in the context of word j, and f is a weighting function that assigns relatively lower weight to rare and frequent co-occurrences.

3.2.3 Paragram

Paragram [31] is another embedding that uses compositional models that can encode arbitrary word sequences into a vector with the property that sequences with similar meaning have high cosine similarity, and that can, importantly, also transfer easily across domains. The method considers six compositional architectures based on neural networks and trains them on noisy phrase pairs from the Paraphrase Database [12].

3.3 LLM with Explicit Semantic Analysis

One special word comparison metrics we used is the Explicit Semantic Analysis (ESA), which was introduced in [11] and uses Wikipedia as its source of world knowledge.

ESA was originally introduced to measure semantic relatedness between text fragments. Given a text fragment, the ESA algorithm generates a set of concepts that are weighted and ordered by their relevance to the input. Here, we provide a brief summary of this approach and refer the reader to

[11] for more details.

The main assumption is that each article in Wikipedia corresponds to a concept. To get the ESA representation of a word, the interpreter identifies the concepts that contain it. These concepts are combined to form a weighted vector, where the weights are obtained by using the TFIDF representation of the original text. The list of concepts is ordered by the weight to get the final ESA representation.

Since Wikipedia is the largest encyclopedic source of knowledge on the web, ESA representation is sufficient for many categorization tasks. Additionally, since Wikipedia was generated by humans, it provides a natural measure of relatedness between text fragments. Previous research has shown that semantic interpretation based on Wikipedia is a more reliable measure of distance between documents than the traditional bag-of-words approach.

CHAPTER 4

SEMANTIC-UNITS TOKENIZATION

The inspiration to our new implementation is to compare different type of tokens using their corresponding metrics. So the most important question is how to tokenize the sentences into different semantic tokens: word, phrases and name-entity.

In this chapter, we propose our approach to accomplish this goal. We first do phrase collection and mining, both from the corpus and knowledge bases. To process input sentences, we then apply named entity recognition method to extract named entities. And we reformat each sentence by concatenating words that belong to same phrases or named entities, with types specified. And we use CogCom Tokenizer package to find all the units at the last step.

4.1 Phrase Mining and Collection

To obtain quality phrases, we extract phrases both from Corpus and knowledge bases. Phrases can be divided into two parts, one is general phrases and another is domain-specific terms. For general phrases, there are two kinds: continuous phrases such as “apply for”, discontinuous phrases such as “pick . . . up”. For domain specific terms, they are mainly continuous phrases and defined in a specific domain such as “support vector machine” in computer science domain. To collect these two type phrases, we extract general two-word phrases from WordNet [21] using method defined in [32] and other multi-word terms from the results of AutoPhrase [28] on the generalized corpus, the Wikipedia dump. We form a phrase collection of size around 185236. After that, we use these phrases collection to reformat sentences.

4.2 Identification of Phrase Continuity

We extract general two-word phrases such as “apply for” from WordNet [21], which is an on-line lexical database. It groups words into many sets by similar meaning. Moreover, it provides words collocations such as “car pool”, “eat out”, etc. We can treat these word collocations as high quality phrases. For now, we obtain high quality general phrases but WordNet [21] doesn’t provide us information about these phrases whether its continuous phrases or discontinuous phrases. Thus, we have to do one more step to distinguish phrases belong to which kind, continuous or discontinuous. Generally, we using the same approach of continuity identification defined in [32].

The idea is that most discontinues phrases are separated by an entity such as “take clothes off” and we assume most discontinues phrases are separated within 5 tokens. So based on this heuristics, by finding the frequency of co-occurrences of target words pair at specific words collocation order, we can search and determine discontinues phrases.

For each phrase, we compute $[w_1, w_2, w_3, w_4, w_5]$ where w_i, $1 \leq i \leq 5$, indicates there are w_i occurrences of A and B in that order with a distance of i. We compute these statistics for a corpus consisting Wikipedia. We set the maximal distance to 5 because discontinuous phrases are rarely separated by more than 5 tokens. If w_1 is 10 times higher than $(w_2 + w_3 + w_4 + w_5)/4$, we classify the as continuous, otherwise as discontinuous.

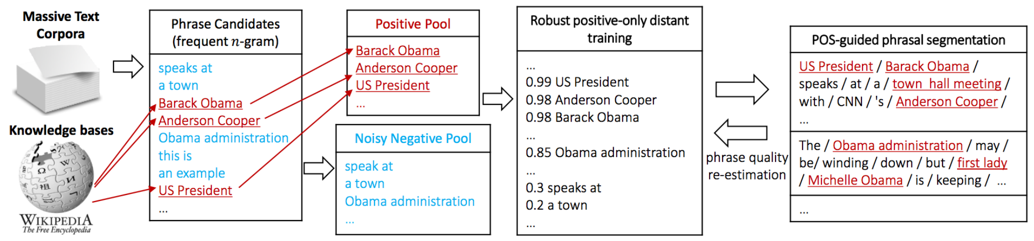
Taking phrase pick off as an example, it gets vector [1121, 632, 337, 348, 4052], w_1 (1121) is smaller than the average 1342.25, so pick off is set as discontinuous. Further consider Cornell University which gets [14831, 16, 177, 331, 3471], satisfying above condition, hence it is treated as a continuous phrase.

4.3 Multi-words Terms

Besides general phrases, named entities and terms are also important parts of phrases. It is hard to obtain those multi-word terms from general dictionaries because most named entities or terms are defined in a specific domain such as ”support vector machine” is an important phrase(term) in computer science domain but it is meaningless in sports or other domains. We employ

AutoPhrase[28] as a helper tool to extract these domain specific terms and named entities.

Figure 4.1: Overview of AutoPhrase [28]


Massive Text Corpora
Knowledge bases
Phrase Candidates (frequent n-gram)
speaks at
a town
Barack Obama
Anderson Cooper
Obama administration
this is
an example
US President
Positive Pool
Noisy Negative Pool
Robust positive-only distant training
0.99 US President
0.98 Anderson Cooper
0.98 Barack Obama
0.85 Obama administration
0.3 speaks at
0.2 a town
phrase quality re-estimation
POS-guided phrasal segmentation
US President / Barack Obama / speaks / at / a / town hall meeting / with / CNN / 's / Anderson Cooper /
The / Obama administration / may / be/ winding / down / but / first lady / Michelle Obama / is / keeping / ...

Comparing with other phrase mining models, AutoPhrase [28] doesn't need human effort to annotate text data as training data. It depends on high quality phrases in knowledge database as positive labels to help generating high quality phrases. Because it contains high quality phrases from knowledge database as positive labels and poor quality phrase candidates as negative labels, the label generation is trustworthy. After obtaining high quality labels, AutoPhrase executes robust positive-only distant training and POS-guided phrasal segmentation. Robust positive-only distant training and POS-guided phrasal segmentation would enhance mutually. The overall framework is shown in 4.1.

4.4 Text Tokenization

After we obtain both the phrases and named entities, we are able to tokenize the raw text into several semantic tokens. This is an essential preprocessing step for comparing semantic units. We are reformatting the sentences, which contains following steps:

Given a sentence "...A B...C...D...E F...",

1. If A and B form a continuous phrase in our collected phrase set and no words between them, we reformat the sentence as "...A_B...C...D...E F..."

2. If C and D form a discontinuous phrase, and they are separated by less than k words, where k is the predetermined threshold. We replace

each of the two words with C_D to make the context of both constituents available to phrase in learning, that is, reformat the sentence as "...A_B...C_D...C_D...E F..."

3. If E and F form a named entity with specific typing in our interested type set, we reformat the sentence as "...A_B...C_D...C_D...E_F:$\langle Type \rangle$..."

CHAPTER 5

LEXICAL SIMILARITY USING SEMANTIC-UNIT COMPARATOR

5.1 Framework

Following the preprocessing method in previous chapter, we can now split a sentence into semantic-unit tokens. For example, the sentence "Trump turn the light off", we reformat it into "Trump(PER) turn_off the light" after semantic-unit tokenization. And for this particular sentence, we now have a phrase "`turn_off`" and a name entity `Donald_Trump(PER)` and word "the, light".

Once we have those different kinds of tokens, we could use corresponding metrics to match the semantic tokens from one sentence to the most similar semantic tokens from the other sentence.

The similarity score is computed by the equation shown below:

$$LLM(s_1, s_2) = \frac{\sum_{t \in \text{tokens}} \sum_{v \in s_1} max_{u \in s_1} sim_t(u, v)}{|s_2|} \tag{5.1}$$

where we use the tokens' corresponding comparison metrics as $sim_t(u, v)$ in the equation. We will talk each different tokens comparison metrics in detail below.

5.2 LLM with Phrase Comparator

The problem of the current uni-gram representation is that it does not have an accurate representation for phrases especially the non-compositional phrase. For example,"support vector machine", combing representation of each word "support" "vector" "machine" fails to capture the semantic meaning of the phrases and thus may lead to token mismatch. In order to accurately compare and match phrases, our first step is to obtain representations for phrases.

5.2.1 Semantic Unit Embedding

It was [19] who really brought word embedding to the forefront through the creation of word2vec, a toolkit enabling the training and use of pre-trained embeddings. Due to the huge success of Word2vec, we decide to use the Skip-gram model in their toolkit to train our semantic unit embedding. We first used text tokenization method described in chapter 4 to process the training corpus so that we have label for both phrases and name-entity. We slightly modified the model object to better fits our semantic unit based embedding. The objective function is modified as below:

$$J_\Theta = \frac{1}{T}\sum_{T}^{t=1}\sum_{-n\leq j\leq n,\neq 0} logp(w_{t+j}, z_{t+j}|w_t, z_{t+j}) \tag{5.2}$$

And the Softmax function is:

$$Pr(\langle w_c, z_c\rangle|\langle w_i, z_i\rangle) = \frac{exp(\mathbf{w}_j^{z_j}\cdot\mathbf{w}_t^{z_t})}{\sum_{\langle w_j, z_j\rangle\in\langle W,T\rangle} exp(\mathbf{w}_j^{z_j}\cdot\mathbf{w}_t^{z_t})} \tag{5.3}$$

where (w, z) is word-phrases pair or name entity-type pair and $\mathbf{w}^z$ is the vector we learn when we regard each pair as a pseudo word.

Once we have the representation, we can compute the cosine as degree similarity between those two semantic units as we did on single words.

5.3 LLM with Name Entity Comparator

One challenge we encounter here is that we can only have embedding representations for very common names. For generalized name and entity, we decide to employ NESim metric [8].

5.3.1 NESim

As with words, named entities need to be compared in a variety of NLP tasks, such as entity/schema matching and named co-reference discovery. For example, in the schema matching task, it is important to know that George Bush is the same as Bush, George or that Mr. Smith is different from Mrs. Smith. Several named entity metrics were developed incorporating inputs from statistical methods, databases, or artificial intelligence [4]. However, most of the existing approaches are limited in two aspects:

1. they do not take advantage of the named entity types when computing the similarity,
2. they do not consider the semantics of the tokens in named entities.

We have found that the types and the semantics of the tokens in named entities play important roles in named entity metrics. For example, George Washington and Washington are similar if we know that they are two person names, but different if they are two locations, and more obviously, they are different if one refers to a location and the other refers to a person. Similarly, it would be simple to compare these two names if we know that George is the first name and Washington is the last name. To address these two limitations, we incorporate the two main improvements specified below.

1. Leveraging the types of named entities in measuring their similarity. Named entity types are given by many named entity recognition packages. Standard types include person, location, and organization. If two names have different types, they should not be considered as similar. Therefore, our similarity computation depends on the named entity types. If two names are labeled as persons, they will be compared based on their identified first names, last names, and if available, their middle names. We also consider replacing nicknames with their original names in order to improve the coverage of our metric (e.g. Bob and Robert). Honorifics are also separated and identified, so that gender based comparison can be made. If two names are locations, the metric considers several standard ways of expressing locations, such as using abbreviations (e.g. IL for Illinois, or VN and VNM for Vietnam) and using a country-language look-up table (e.g. Russia and Russian.) For organizations, our metric is able to capture acronyms which are often used when an organization name is mentioned many times (e.g. NATO for North Atlantic Treaty Organization). If the type is not known, the metric tries the three types one-by-one, and returns the highest score.

2. Parsing the input names to identify the semantics of each token in the names. This is required especially for person names. In order to compare first names, last names, and middle names of persons, the metric parses the input names into fields and compares them separately. Person names are parsed using common cues in name format, such as names with or without commas (for first and last names), names with or without abbreviation (for first and middle names), etc. The metrics parser also identifies organization acronyms by combining the initial letters of name tokens. It is worth noting that there are several cases where this heuristic is not sufficient to form an acronym (e.g. AIRTC stands for Air Training Corps). The parser deals with this phenomenon by simply trying several combinations of the name tokens first letters. As a final back-off step, our metric uses edit distance metrics (viz. the one proposed by [4]) to measure the similarity between named entities if none of the above conditions is matched. Table 1 shows some similarity scores of the input names given by a metric using Jaro-Winkler distance (JRWK) [4]) and our method, NESim.

CHAPTER 6

EXPERIMENTS

In this chapter, we conduct experiments to show the effectiveness of semantic-unit matching against the traditional word-unit matching.

6.1 Datasets Description

The first task we consider is the paraphrase or semantic relatedness identification. To evaluate the performance of paraphrase identification, we use the Microsoft Research Paraphrase Corpus (MSRP) [9] dataset. In this dataset, two sentences are given and we are expected to predict whether or not they are paraphrases. The training set consists of 4076 sentence pairs (2753 which are positive) and the test set has 1725 pairs (1147 are positive).

The other dataset is the SemEval 2014 Task 1: semantic relatedness SICK dataset [17]. Given two sentences, the goal is to produce a score of how semantically related these sentences are, based on human generated scores. Each score is the average of 10 different human annotators. Scores take values between 1 and 5. A score of 1 indicates that the sentence pair is not at all related, while a score of 5 indicates they are highly related. The dataset comes with a predefined split of 4500 training pairs, 500 development pairs and 4927 testing pairs. All sentences are derived from existing image and video annotation datasets.

6.2 Comparison Metric

For Microsoft Paraphrase task, We compute the snippet similarity for all 5801 pairs of the corpus, using the different measures defined in chapter 2 and 4: LLM with word comparator and LLM with semantic-unit comparator. After finding the similarity scores using the similarity metrics, we rank the documents on the similarity score and choose a threshold that maximizes the accuracy over the training data. We report the accuracy scores over the complete dataset. Accuracy measures the fraction of all instances that were labeled correctly, including both positive and negative instances. F1 is the harmonic mean of the precision and recall values. In this evaluation, Accuracy is the more appropriate measure, since it is important to recognize both positive and negative instances correctly.

For the semantic relatedness SICK dataset, we compute the similarity score, using the different measures defined in chapter 2 and 4, between all the pairs of the sentences. And then we report the Pearson's r correlation between the similarity score and human annotation score.

6.3 Experimental Results

Table 6.1: Accuracy on Microsoft Paraphrase Identification Corpus and Pearson's r Correlation on SICK Relatedness Dataset

	MSRP Corpus	**SICK Dataset**
LLM with Wordnet	**0.716**	**0.6728**
LLM with Word2vec	**0.706**	**0.637**
LLM with GloVe	**0.694**	**0.649**
LLM with Phrases(Wordnet)	**0.718**	**0.673**
LLM with Phrases(Wordnet) and NER	**0.705**	**0.656**
LLM with Phrases Embedding	**0.709**	**0.632**
LLM with Phrases Embedding and NER	**0.7**	**0.625**
Skip-thought[14]	**0.758**	**0.7995**
compositional embedding [20]	**0.73**	*
sentence representation[5]	*	**0.84**

In the table, the leftmost column is the name of the method we compared. The number in the middle column is accuracy score on MSRP and the number in the rightmost column is Pearson's r Correlation reported on SICK Dataset by each metric.

The "LLM with phrases Embedding" means LLM using phrase embedding representation as the matching comparator. Since the representation is trained with skip-gram model. So comparing with "LLM with word2vec", "LLM with phrase" has very slight improvement on Microsoft paraphrase test. The "LLM with phrases(Wordnet)" means LLM using phrases as tokens and use Wordnet to compare them. As we can see, phrases tokenization is more effective than the simple unigram tokenization.

"LLM with phrase Embedding and NER" means LLM with both phrase embedding and NE comparator. "LLM with phrase(Wordnet) and NER" means LLM using Wordnet to compare phrase units and also using the NE comparator. But unfortunately, NE comparator doesnt improve much on the results as we expected. One reason is that both datasets have very few sentences has person's or location name. Another possible cause is that NESim metric gives us very low score on the different name and thus lead to low sentence similarity score on the similar sentences pair. For example, "James turn off the light" and "John turn off the light", the two sentence are supposed to be very similar. But NESim returns 0 score for the two name "James" and "John" and leads to low sentence similarity score in overall.

For the Microsoft paraphrase, "Wordnet" metric is close to the-state-of-art result by the unsupervised method reported by [20]. For the sick dataset, the results we have are not very promising compared with state-of-the-art results by neural sentence representation model [5][14].

6.4 Case Study

We present some cases to demonstrate the effectiveness of LLM with semantic-unit comparator in this section.

6.4.1 Phrase Similarity

In table 6.2, we find that Skip-gram doesn't have an accurate representation of the the non-compositional phrase "support vector machine". In contrast, with our phrase embedding, we can handle this non-compositional phrase very well and the model finds us really similar terms.

Table 6.2: Selected Most Similar Words/Semantic Units of "Support Vector Machine".

Rank	Skip-gram	Semantic Unit Based
# 1	vector	SVM
# 2	matrix	discriminative classifiers
# 3	scalar	kernel-based
# 4	learning	classification

6.4.2 Phrases Comparator

In unigram matching model, it's unlikely that we will have high similarity score between the two sentences:

1. "He turns the light on."
2. "He opens the light."

Since the unigram matching model doesn't have representation for phrase "turns_on". The mismatching of phrase "turn on" and "open" will cause very low sentence similarity score. While with our phrase comparator, the sentence will be reformatted and move word "turns" and "on" together as "turns_on". Then we will perfectly match "turns_on" and "opens" since they have high similarity score in the phrase representation.

6.4.3 Name Entity Comparator

Without name entity comparator, the similarity between the two sentences:

1. "Washington was not at Boston."
2. "Boston is not in Washington State"

are likely to be very high. Since traditional LLM implementation does not distinguish the person Washington and the location Washington. The alignment will just simply match the word "Washington" from both sentences. While in our approach, we have NER annotator to tag name-entity with its typing for above terms. So we can reformat sentences into

1. "Washington:PER was not at Boston."
2. "Boston is not in Washington_State:LOC"

And with the NESim metric, the new model will definitely not match those tokens and reports an accurate similarity score.

6.4.4 Phrases and Name Entity Comparator

Considering the following examples:

1. "He likes playing computers in Siebel Center"
2. "Sibel likes the computer science major"

The traditional LLM with match the same word "Siebel" and "Computer" in both sentences but mismatch the other words.

With only phrase comparator, the sentence will be reformatted into:

1. "He likes playing computers in Siebel Center"
2. "Sibel likes the computer-science major"

and still match the same word "Siebel".

With both phrases and name entity comparator, the sentence will become:

1. "He likes playing computers in Siebel-Center:LOC"
2. "Sibel:PER likes the computer-science major"

and thus it will accurately report the similarity score.

So as we can see, with phrases and NER, it can greatly help us to infer correctly that the sentences are paraphrases or related. Hence, our model seems very promising in those tasks.

CHAPTER 7

CONCLUSION

In my thesis work, we first summarized the approach of lexical similarity, phrase mining, name entity recognition and current LLM implementation. We extend LLM word comparator with many word similarity metrics. We obtain an embedding representation for generalized phrases. And we use the phrases dictionary we built along with NER annotator to chunk the raw text into semantic tokens: word, phrases, name-entity. Then we propose our new lexical level matching method which matching semantic tokens instead of word tokens between two sentences. Our method extends lexical matching from word-based to semantic-unit based.

Our experiments show that semantic unit based matching outperforms word level based matching on multiple datasets.

7.1 Future Work

Compared with the-state-of-art sentence representation method, our approach didn't show very promising results. We could try the more complicated method in the matching process. For example, the attention module in neural machine translation is very similar to our matching process and it may work better than the simple similarity metrics. The basic mechanism could be encoding both sentences and use attention module to match tokens in their decoding process.

What's more, we could try to apply our model on other NLP tasks to further prove the robustness of our model. Since most NLP application is dealing with the single word, our model is changing the basic unit from single word token to semantic-unit tokens. We believe this model can be generalized to many tasks and it may achieve a very promising result intuitively.

Cross-lingual NLP is a very applicable field. We could use cross-embedding to match cross-language sentences. Cross-embedding representations have already been useful and proven to be successful in a variety of tasks including similarity tasks. So it will be very interesting to see what we can do with LLM using cross-lingual embedding similarity metrics.

REFERENCES

[1] C. C. Aggarwal and C. Zhai, *Mining text data.* Springer Science & Business Media, 2012.

[2] R. M. Aliguliyev, “A new sentence similarity measure and sentence based extractive technique for automatic text summarization,” *Expert Systems with Applications*, vol. 36, no. 4, pp. 7764–7772, 2009.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model,” *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[4] W. Cohen, P. Ravikumar, and S. Fienberg, “A comparison of string metrics for matching names and records,” in *Kdd workshop on data cleaning and object consolidation*, vol. 3, 2003, pp. 73–78.

[5] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, “Supervised learning of universal sentence representations from natural language inference data,” *arXiv preprint arXiv:1705.02364*, 2017.

[6] M. Connor and D. Roth, “Context sensitive paraphrasing with a global unsupervised classifier,” in *ECML*, vol. 7. Springer, pp. 104–115.

[7] P. Deane, “A nonparametric method for extraction of candidate phrasal terms,” in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics.* Association for Computational Linguistics, 2005, pp. 605–613.

[8] Q. Do, D. Roth, M. Sammons, Y. Tu, and V. Vydiswaran, “Robust, light-weight approaches to compute lexical similarity,” *Computer Science Research and Technical Reports, University of Illinois*, p. 94, 2009.

[9] B. Dolan, C. Quirk, and C. Brockett, “Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources,” in *Proceedings of the 20th international conference on Computational Linguistics.* Association for Computational Linguistics, 2004, p. 350.

[10] K. Frantzi, S. Ananiadou, and H. Mima, “Automatic recognition of multi-word terms:. the c-value/nc-value method,” *International Journal on Digital Libraries*, vol. 3, no. 2, pp. 115–130, 2000.

[11] E. Gabrilovich and S. Markovitch, "Computing semantic relatedness using wikipedia-based explicit semantic analysis." in *IJcAI*, vol. 7, 2007, pp. 1606–1611.

[12] J. Ganitkevitch, B. Van Durme, and C. Callison-Burch, "Ppdb: The paraphrase database." in *HLT-NAACL*, 2013, pp. 758–764.

[13] N. Kaji and M. Kitsuregawa, "Building lexicon for sentiment analysis from massive collection of html documents." in *EMNLP-CoNLL*, 2007, pp. 1075–1083.

[14] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Advances in neural information processing systems*, 2015, pp. 3294–3302.

[15] O. Levy, Y. Goldberg, and I. Dagan, "Improving distributional similarity with lessons learned from word embeddings," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 211–225, 2015.

[16] J. Liu, J. Shang, C. Wang, X. Ren, and J. Han, "Mining quality phrases from massive text corpora," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 2015, pp. 1729–1744.

[17] M. Marelli, L. Bentivogli, M. Baroni, R. Bernardi, S. Menini, and R. Zamparelli, "Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment," *SemEval-2014*, 2014.

[18] R. Mihalcea, C. Corley, C. Strapparava *et al.*, "Corpus-based and knowledge-based measures of text semantic similarity," in *AAAI*, vol. 6, 2006, pp. 775–780.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[20] D. Milajevs, D. Kartsaklis, M. Sadrzadeh, and M. Purver, "Evaluating neural word representations in tensor-based compositional settings," in *In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP*. Citeseer, 2014.

[21] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[22] B. Mohit, "Named entity recognition," in *Natural Language Processing of Semitic Languages*. Springer, 2014, pp. 221–245.

[23] A. Parameswaran, H. Garcia-Molina, and A. Rajaraman, "Towards the web of concepts: Extracting concepts from large datasets," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 566–577, 2010.

[24] T. Pedersen, S. Patwardhan, and J. Michelizzi, "Wordnet:: Similarity: measuring the relatedness of concepts," in *Demonstration papers at HLT-NAACL 2004*. Association for Computational Linguistics, 2004, pp. 38–41.

[25] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation."

[26] V. Punyakanok and D. Roth, "The use of classifiers in sequential inference," in *Advances in Neural Information Processing Systems*, 2001, pp. 995–1001.

[27] L. Ratinov and D. Roth, "Design challenges and misconceptions in named entity recognition," in *CoNLL*, 6 2009. [Online]. Available: http://cogcomp.cs.illinois.edu/papers/RatinovRo09.pdf

[28] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han, "Automated phrase mining from massive text corpora," *arXiv preprint arXiv:1702.04457*, 2017.

[29] D. Wang, T. Li, S. Zhu, and C. Ding, "Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2008, pp. 307–314.

[30] R. Wang and G. Neumann, "Recognizing textual entailment using sentence similarity based on dependency tree skeletons," in *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. Association for Computational Linguistics, 2007, pp. 36–41.

[31] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, "Towards universal paraphrastic sentence embeddings," *arXiv preprint arXiv:1511.08198*, 2015.

[32] W. Yin and H. Schütze, "An exploration of embeddings for generalized phrases." 2014.

[33] Z. Zhang, J. Iria, C. Brewster, and F. Ciravegna, "A comparative evaluation of term recognition algorithms," 2008.