Conceptualization of Sentence Paraphrase Recognition with Semantic Role Labels

R. Yadav¹, A. Kumar¹, A. Vinay Kumar², and P. Kumar¹
¹Information Technology & Systems, Indian Institute of Management, Lucknow, Uttar Pradesh, India
²Finance & Accounting, , Indian Institute of Management, Lucknow, Uttar Pradesh, India

Abstract - Sentence paraphrase recognition plays an important role in many NLP applications. In majority of previous studies, basic unit of information for analysis is the sentence itself. However, a sentence contains information about one or more events/entities in its multi-clausal structure. Clauses – conceptualized as Semantic Role Labels (SRL) or predicate-argument tuples – are the smallest grammatical units with which sentence information can be comprehended and compared with. Objective of this paper is to propose a sentence paraphrase recognition methodology using predicate-argument tuples as the basic unit of information.

This paper introduces concept of paraphrasing, loosely paired, and unpaired tuples to establish sentence-sentence similarity. Two sentences are paraphrasing if they contain at least one paraphrasing tuple; and no or insignificant dissimilarities (loosely paired and unpaired tuples). The paper proposes two tuple representation schemes – first Vector Space Model based, and second based on distributed word representations (embeddings) learnt using deep Neural Network language models.

Keywords: Paraphrase Recognition, Semantic Role Labels, Predicate argument, Vector Space Model, Recursive Auto-Encoders

1 Introduction

Any two natural language expressions are called paraphrase (para- ‘expressing modification’ + phræzaein ‘tell’) if both convey similar information or meaning. Sentence paraphrase recognition is essentially a boolean sentence-sentence similarity metric that is indispensable to many Natural Language Processing (NLP) applications like question-answering [1], text summarization [2], machine translation [3] etc. Understanding a sentence – with all its possible deep syntactic structure variations, language semantic nuisances like synonyms, idioms etc. – has been a challenging task. In majority of previous studies ([4], [5], [6], [7]), basic unit of analysis for sentence comprehension is the sentence itself. However, a sentence is generally a multi-clause grammatical structure – referred to as Semantic Role Labels (SRL) or predicate-argument tuples – conveying information about more than one event/entity at a time. SRLs or predicate-argument tuples are the smallest grammatical unit of information that can be used to comprehend sentence meaning [8]. Objective of this study is to propose a sentence paraphrase recognition methodology with SRLs as its basic unit of analysis.

Defined for each instance of sentence’s predicate (verb), semantic role labeling entails assigning role of WHO did WHAT to WHOM, WHEN, WHERE, WHY, HOW etc. according to predicate’s verb frame [9] – collectively called predicate and its corresponding arguments. This paper uses the term predicate-argument tuple or just tuple interchangeably to refer to predicate and its argument SRLs. A sentence S having m predicates is set of m predicate-argument tuples:

\[ S = \{PA_1, PA_2, ..., PA_m\} \]

where, \( PA_i = \{p_i, a_{i0}, a_{i1}, ..., a_{iK}\} \) with \( 1 \leq i \leq m \)

Here, K is the size of domain of arguments labeled by a Semantic Role Labeler ([8], [9]).

For instance, sentence example \( S_{ex1} \) – "Amrozi accused his brother, whom he called “the only witness”, of distorting his evidence." – has three predicates – accused, called, and distorting.

Amrozi\(_{\{p1a0\}}\) accused\(_{\{p1a1\}}\) (his brother)\(_{\{p1a2\}}\) ... (whom)\(_{\{p2a0\}}\) he\(_{\{p2a1\}}\) called\(_{\{p2a2\}}\) "(the only witness)\(_{\{p3a0\}}\) ... of distorting\(_{\{p3a1\}}\) (his evidence)\(_{\{p3a2\}}\) ...

Hence, sentence \( S \) can be represented as set of three predicate argument tuples \( (m = 3) \) as depicted in Table I.

<table>
<thead>
<tr>
<th>T.I.</th>
<th>p</th>
<th>a0</th>
<th>a1</th>
<th>a2</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA1</td>
<td>accused</td>
<td>Amrozi</td>
<td>his brother</td>
<td>of distorting his evidence</td>
</tr>
<tr>
<td>PA2</td>
<td>called</td>
<td>he</td>
<td>whom, the only witness</td>
<td></td>
</tr>
<tr>
<td>PA3</td>
<td>distorting</td>
<td>his brother</td>
<td>his evidence</td>
<td></td>
</tr>
</tbody>
</table>

In a well written sentence, \( m \) will generally be less than five. Microsoft Research (MSR) Paraphrase Recognition Corpus [10] has on an average \( m = 2.24 \) [11]. For comparing two sentences \( S_1 \) and \( S_2 \), each tuple of sentence needs to be compared with each tuple of another sentence.

From all the possible pairings \( S_1 X S_2 \), Qiu et al [11] defined two categories of tuples – semantically paired tuples and unpaired tuples. This paper extends it to following three categories of possible pairings:
1. **Semantically paired tuples or Paraphrasing tuples** – These tuples convey similar meaning about same event or same entities of a sentence pair.

2. **Loosely paired tuples** – These tuple pairs are responsible for conveying same part of information content of each sentence. They may talk about same event or same actors, but they do not convey same meaning. Loosely paired tuples help in identifying unpaired tuples that have no counter-part in the other sentence.

3. **Unpaired tuples** – Tuples that are neither loosely paired nor semantically paired are unpaired tuples of a sentence pair.

It is assumed that the given sentences are from same context [12]. Concept of pairing is elucidated below with the help of three hypothetical sentences:

- **S₁**: Amrozi accused his brother, whom he called “the only witness”, of distorting his evidence.
- **S₂**: Amrozi accused his brother of deliberately altering his evidence.
- **S₃**: Referring to him as a liar, Amrozi accused his brother of deliberately distorting his evidence.

**S₁** has three predicates = \{accused, called, distorting\}; **S₂** has two predicates = \{accused, altering\}; **S₃** has three predicates = \{Referring, accused, disturbing\} as shown in Table II.

<table>
<thead>
<tr>
<th></th>
<th>T₁</th>
<th>p</th>
<th>a₁</th>
<th>a₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>PA₁₁</td>
<td>accused</td>
<td>Amrozi</td>
<td>his brother</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>of distorting his evidence</td>
</tr>
<tr>
<td>S₂</td>
<td>PA₁₂</td>
<td>called</td>
<td>He</td>
<td>whom, the only witness</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S₃</td>
<td>PA₁₃</td>
<td>distorting</td>
<td>his brother</td>
<td>his evidence</td>
</tr>
<tr>
<td>S₁</td>
<td>PA₂₁</td>
<td>accused</td>
<td>Amrozi</td>
<td>his brother</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>of deliberately distorting his evidence</td>
</tr>
<tr>
<td>S₂</td>
<td>PA₂₂</td>
<td>altering</td>
<td>his brother</td>
<td>his evidence</td>
</tr>
<tr>
<td>S₃</td>
<td>PA₃₁</td>
<td>referring</td>
<td>Amrozi</td>
<td>him as a liar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S₃</td>
<td>PA₃₂</td>
<td>accused</td>
<td>Amrozi</td>
<td>his brother</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>of deliberately distorting his evidence</td>
</tr>
</tbody>
</table>

Comparing sentences **S₁** and **S₂**, \((PA₁₁, PA₁₂)\) and \((PA₁₃, PA₂₂)\) are semantically paired while **PA₁₂** is unpaired and insignificant to its sentence meaning – hence **S₁** and **S₂** are paraphrase. Comparing sentences **S₁** and **S₃**, \((PA₁₁, PA₁₃)\) and \((PA₁₃, PA₃₂)\) are semantically paired while \((PA₁₂, PA₃₂)\) are loosely paired and significant to the meaning of two sentences – hence **S₁** and **S₃** are not paraphrase. Hence, a sentence pair is paraphrasing if it contains semantically paired tuples and has no or insignificant dissimilarities [11]. Sentence-sentence similarity can be established in terms of semantically paired or paraphrasing tuples while dissimilarities in terms of loosely paired and unpaired tuples.

Qiu et al [11], in their work on paraphrase recognition using SRLs, decomposes sentence paraphrase recognition task into two predicate-argument tuple level tasks – first is semantically paired tuples identification heuristic and second is unpaired tuple significance classification. Authors represented SRLs with their respective syntactic headword [9] and compared them using Lin’s thesaurus similarity metric [13]. Similarity between two tuples is established using a weighted average of similarity between their predicate and argument labels. Sentence pair similarities and dissimilarities are identified heuristically in terms of semantically paired tuples and unpaired tuples respectively. Two sentences are paraphrasing if any dissimilarity (unpaired tuples) present is insignificant. Qiu et al [11] derives tuple significance training data set from MSR paraphrase recognition data set [10] to learn dissimilarity significance classification. Authors [11] reported recall of 0.934 and precision of 0.725. Low precision is attributed mainly to paraphrase recognition failure in case of complex multi clause sentences [11]. This could be primarily because authors approximate SRL phrases with their syntactic headword which risk losing significant information in case of long SRL phrases.

This paper proposes two predicate-argument tuple representation schemes – first, Vector Space Model (VSM) based representation; and second, deep Neural Network language model based word and phrase embeddings (14, 15, 16). For comparing two tuples, tuple paraphrase recognition is learnt as a separate classification task with tuple-tuple similarity matrix as the feature set. Like Qiu’s work [11], this paper formalizes sentence level paraphrase recognition into two tuple level classification tasks – first, tuple level paraphrase recognition and second dissimilarity significance classification. In order to derive training data sets for these two tuple based classification tasks from sentence based MSR paraphrase corpus [10], concept of loosely paired tuples is introduced. Loosely paired tuples discuss same event or same entity in a sentence pair, but are not semantically similar or paraphrasing. This category of tuple pairs helps in deriving negative examples for tuple paraphrase recognition data set and also helps in refining and enhancing unpaired tuple significance classification data set.

The paper presents an SRL based paraphrase recognition approach in following sections. Section 2 gives an overview of SRL based representation schemes adopted in past and characteristics of an efficient representation scheme needed for paraphrase recognition task. It further proposes two SRL representation schemes - Vector Space Model (VSM) based representation; and second, deep Neural Network language model based word and phrase embeddings (14, 15, 16). Overall sentence paraphrase recognition methodology is
discussed in section 3. This is followed by conclusion and future directions in Section 4.

2 Semantic Role Labels Representation

To the best of our study and knowledge, Qiu’s work has been the only work that deploys SRLs to represent and compare two sentences. Qiu et al [11] reduced each SRL phrase to its corresponding syntactic headword feature and used Lin’s thesaurus based word-word similarity measure as θ.

Reducing an SRL phrase with its syntactic headword often lose the main phrase content or the words modifying meaning of the phrase content. For instance, for sentence S

\[
S_{ex2}: \text{Revenue in the first quarter of the year dropped 15 percent from the same period a year earlier.}
\]

Considering arguments for predicate “drop”, syntactic head word \(a_1\) noun phrase “revenue in the first quarter of the year” is reduced to “year” while syntactic head word for \(a_2\) prepositional phrase “from the same period a year earlier” is reduced to “period” (see Error! Reference source not found.). It is observed that approximating a phrase with its syntactic headword may risk losing information significant in paraphrase recognition.

Qiu [11] used Lin’s thesaurus based word-word similarity measure for comparing two phrases headwords. Lin thesaurus [13] major drawback is consideration of antonyms and unrelated words as proximate neighbours of a word. Further, Lin thesaurus [13] similarity measure between two words is independent of sentences’ context or discourse and hence sentence context has no role to play in disambiguating accurate sense of a word.

Predicate-argument tuple, \(PA = \{p, a_0, a_1, ..., a_K\}\) is an ordered collection of a sentence’s predicate phrase (verb/event) and its corresponding argument label phrases. With number of argument types \(K\) fixed, tuple representation is essentially a function of a phrase representation scheme \(\psi\)

\[\psi_{VSM}\]

such that, given an appropriate similarity metric 0, similar phrases have high similarity and dissimilar phrases have low dissimilarity.

In MSR paraphrase corpus [10], each sentence has on an average 2.25 tuples, and hence comparing two sentences needs on an average approximately five tuple-tuple comparisons. Hence, an efficient tuple or phrase representation scheme \(\psi\), given an appropriate choice of 0, should facilitate a fast tuple-tuple comparison. Lexical string based phrase representations needs elaborate string to string comparison metric and hence cannot support a fast tuple-tuple comparisons. This paper suggests two vector based tuple representation schemes. First representation scheme \(\psi_{VSM}\) is based on Vector Space Model (VSM) with binary weights – signifying presence or absence of a feature. Second representation scheme \(\psi_{BASE}\) is based on deep Neural Network Language Model trained word and phrase embeddings ( [14], [15], [16]).

2.1 Vector Space Model based SRL representation scheme

First representation scheme \(\psi_{VSM}\) is based on Vector Space Model (VSM) with binary weights – signifying presence or absence of a feature. Feature definition for \(\psi_{VSM}\) is lemmatized content words (noun, verb, adjective, adverbs). Features are normalized with numbers and Named Entity (person, location, percentage, currency, title, company) based abstractions (see Error! Reference source not found.). Pronouns are treated as wildcards for all possible named entities of the corresponding sentence pair. Feature vocabulary \(V_S\) is local to a given sentence pair, where each phrase of that sentence pair can be represented with a \(|V_S|\) size binary vector. Suggested choices of similarity metric 0 are cosine similarity metric or Jaccard similarity metric.

Since predicate phrase is one of the most important element of a tuple in paraphrase recognition [11] and verbs being one of the most polysemous in nature, it is important to incorporate a verb sense disambiguation algorithm for an improved recall. This paper implements a verb disambiguation algorithm based on Galley’s [17] linear order lexical chain based noun disambiguation algorithm. Author [17] scans text to identify candidate words (nouns) while simultaneously creating a disambiguation graph where all words are attached with weighted edges with respect to following semantic relations – synonym, hyponym, hypernym and coordinate words. This paper scans a sentence pair for candidate verbs (and possible verb nominalizations [18]) creating a graph where all verbs are attached with positive weighted edges with respect to following semantic relations – synonym, hypernym, entailment and coordinate verbs; and with negative weighted edges for antonym relations. On testing the algorithm on around 200 MSR paraphrase sentence-pairs, it is able to detect verb relation between sentence-pair instances like “…share were up…”, “…shares jumped…”, “…shares rose…” or “…shares increased…” successfully. It is asserted that proposed verb

### Table III
SYNTACTIC HEADWORD AND VSM BASED TUPLE REPRESENTATION SCHEMES FOR \(S_{ex2}\)

<table>
<thead>
<tr>
<th>Tuple</th>
<th>Predicate ((p))</th>
<th>(Arg0 (a_0))</th>
<th>(Arg1 (a_1))</th>
<th>(Arg2 (a_2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA(_d)</td>
<td>dropped</td>
<td>revenue, first, quarter, year</td>
<td>from the same period a year earlier</td>
<td></td>
</tr>
<tr>
<td>Syntaxic Headword</td>
<td>dropped</td>
<td>year</td>
<td>percent</td>
<td>period</td>
</tr>
<tr>
<td>VSM features</td>
<td>drop</td>
<td>revenue, first, quarter, year</td>
<td>(%)NUMBER%</td>
<td>same, period, year, earlier</td>
</tr>
</tbody>
</table>

1 Syntactic head of a phrase calculated using a head word table described in [22], Appendix A with modifications on prepositional phrases proposed by [9]
disambiguation algorithm ought to improve recall for sentence pair recognition task.

### 2.2 Recursive Auto-Encoder based SRL representation scheme

Another area of emerging interest in NLP is use of deep neural language models [19] to learn distributed word representations ([14], [15]) or phrase representations [16] in an unsupervised manner such that these models can be reused in other specific supervised NLP tasks like POS tagging [20], NER recognition [20], paraphrase recognition [16] etc.

Distributed word representation (or embeddings) is a d-dimensional vector such that semantically or syntactically similar words have embeddings closer to each other. This leads to a smoother solution space with lesser discontinuities and hence model trained on such a space will have better generalization on unseen data [14]. The word embedding matrix \( \mathbf{W} \), where \( |V| \) is the size of vocabulary, is learnt jointly as part of an unsupervised deep neural language model ([14], [15]). Such unsupervised neural language models based on distributed word representation are reusable in other NLP tasks.

Using Turian embeddings [15] as word representation scheme, Socher et al [16] introduced auto-encoder – an unsupervised neural network model – that encode a bi-gram embedding of size \( 2 \times d \) into a \( d \times d \) size vector such that the bi-gram can be reconstructed back with minimum reconstruction error. To encode a sentence, this auto-encoder is applied recursively on sentence’s syntactic parse tree in right-to-left bottom-up manner such that all its non-leaf node phrases are encoded into a fixed \( d \times d \) size vector minimizing unfolding reconstruction error at each node – the model referred to as unfolding Recursive Auto-Encoders (RAE) [16]. Once a word embedding matrix \( \mathbf{L} \) and an RAE is learnt on an unlabeled corpus, these can be re-used to encode any sentence’s syntactic parse tree. Socher et al [16] applied unfolding RAE based sentence representation for paraphrase recognition reporting state-of-the-art accuracy of 76.8%.

This paper proposes to use RAE-encoded parse tree for predicate-argument tuple representation. However, a syntactic parser like Stanford parser [21] adheres to grammatical understanding of predicate while SRL literature [8] is based on logical understanding of predicate. From grammatical perspective, a sentence has two components – the subject; and the rest of the sentence part called predicate that modifies the subject. On the other hand, Proposition bank [8] – the annotated dataset used for SRL task – defines predicate as the verb and its related adverbs or auxiliaries modifying the verb; while arguments are the subjects, direct or indirect objects of the predicate defined as per corresponding verb frame [8].

Passing sentence syntactic parse tree to Socher’s RAE will encode each intermediate node of the tree in a fixed size vector. However, phrases corresponding to predicates are not preserved with these nodes. This difference is best elucidated with following example sentence and its parse tree (Fig. 1).

**Fig. 1. Syntactic Parse Tree for Sentence S**

S: “Amrozi denied to accuse his brother.”

Hence parse tree needs to be transformed at every verb phrase such that its logical predicate phrase and its arguments are preserved in its intermediate nodes as shown in Fig. 2.

**Fig. 2. Transformed syntactic parse tree so that predicate SRL phrase is preserved**

However, Socher’ RAE is trained on Stanford parser’s syntactic parse tree and one needs to verify whether the RAE gives same quality of encodings with transformed parse tree too i.e. with no significant increase in reconstruction error. This was verified on 200 sentence-pair sample taken from MSR paraphrase corpus. The change in reconstruction error of a sentence ROOT node encoding was observed to be insignificant with \( p \)-value 0.021. This verifies that Socher’s RAE can be safely used with transformed trees preserving predicate phrases.

Fixed size encodings of the parse tree thus extracted are used for predicate-argument tuple representation. Hence each tuple can be encoded in \((K+1) \times d\) size matrix where \( K \) is the number of argument SRLs and \( d \) is the size of word embeddings used in Socher’s RAE. Suggested choice for comparing two phrases is Euclidean distance.
3 Methodology

In previous section, two SRL representation schemes \( \psi_{\text{VSM}} \) and \( \psi_{\text{RAE}} \) were proposed where each SRL is represented as \( |V_s| \) size and \( d \) size vector respectively, where \( |V_s| \) is the size of local sentence-pair’s unique feature set while \( d \) is the size word embeddings. Choice of \( 0 \) proposed for \( \psi_{\text{VSM}} \) is cosine similarity and Jaccard measure while choice of \( 0 \) proposed for \( \psi_{\text{RAE}} \) is Euclidean distance metric. Let the choice of SRL representation scheme and similarity measure be \( \psi \) and \( 0 \) in general.

3.1 Tuple-Tuple Similarity Matrix

Given two sentences \( S_1 \) and \( S_2 \) having \( m \) and \( n \) predicate-argument tuples respectively, and \( PA_{i1} \) and \( PA_{i2} \) are the \( i^{th} \) and \( j^{th} \) tuple from \( S_1 \) and \( S_2 \):

\[
S_1 = \{PA_{11}, PA_{12}, ..., PA_{1m}\} \quad \text{and} \quad S_2 = \{PA_{21}, PA_{22}, ..., PA_{2n}\}
\]

\[
PA_{i1} = \{p_1, a_{10}, a_{11}, ..., a_{1k}\}; \quad PA_{i2} = \{p_2, a_{20}, a_{21}, ..., a_{2l}\}
\]

Where, \( K \) is the size of domain of arguments labelled by Semantic Role Labeller. A predicate generally has two to four arguments \([8]\) and hence maximum arguments will be \( \text{null} \). Each SRL is encoded using SRL representation scheme \( \psi \).

For comparing two sentences, one need to consider all possible tuple comparisons to find semantically paired, loosely paired and unpaired tuples. For comparing any two tuples \( PA_{i1} \) and \( PA_{i2} \), this work proposes to use a similarity matrix similar to Socher’s \([16]\) work using similarity metric \( 0 \). Matrix is subsequently normalized such that each entry lies between zero and one. Unlike Socher’s work, pooling is now already defined where each SRL element of a tuple is pooled into one region. Similarity of each pooled rectangular region is calculated using max operator (\( \text{min} \) operator in case of Euclidean distance metric). Resulting pooled \((K+1) \times (K+1)\) matrix can be fed to a classifier for learning paraphrasing characteristics.

Tuple-tuple similarity matrix thus created not only accounts for element-wise similarity but also captures cross SRL alignment between two tuples. For instance in following sentence pair:

\[ S_1: \text{Troy is sentenced to life in prison without parole.} \]

\[ S_2: \text{Troy face life sentence in prison without parole.} \]

\[ PA_{11} = \{[p, \text{“sentenced”}], [a_{10}, \text{null}], [a_{11}, \text{“Troy”}], [a_{20}, \text{“to life in prison”}], ..., [a_{1k}, \text{“without parole”}], ...\} \]

\[ PA_{12} = \{[p, \text{“face”}], [a_{10}, \text{“Troy”}], [a_{11}, \text{“life sentence”}], ..., [a_{20}, \text{“in prison”}], [a_{21}, \text{“without parole”}], ...\} \]

Here, predicates for \( PA_{11} \) and \( PA_{12} \) are “sentence” and “face” respectively. Both these verbs follow different verb frames and hence argument \( a_1 \) of “sentence” is argument \( a_0 \) of “face”, argument \( a_2 \) of “sentence” is argument location of “face”, etc. Similarity matrix captures verb-frame alignment for these two verbs efficiently as shown in Table IV (with darker shade signifying higher similarity).

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{without} & a_{max} & \text{null} & a_{null} & \text{to life in prison} \\
\hline
\text{null} & a_{null} & \text{null} & a_{null} & \text{null} \\
\hline
\text{Troy} & a_{1} & a_{1} & a_{2} & a_{loc} \\
\hline
\text{sentenced} & p & a_{2} & a_{1} & a_{loc} \ \\
\hline
\end{array}
\]

Further, the predicate-argument tuple vectors represented with a predicate and its \( K \) argument type phrases ought to have majority of its elements \( \text{null} \) leading to a sparse tuple-tuple similarity matrix. Instances of comparison of a \( null \) SRL with non \( null \) and \( null \) SRLs need to be handled separately as both cases hold different information for tuple-tuple comparison.

3.2 Sentence Paraphrase Recognition Training

\( \psi \) and \( 0 \), sentence level paraphrase recognition task is divided into two phases – first, learning tuple level paraphrase recognition classification \( \xi \) and second learning dissimilarity significance classification \( \bar{\xi} \). Hence, sentence-pair paraphrase detection training data set needs to be translated to create training data set for paired tuples paraphrase recognition and dissimilarity significance classification tasks.

In phase I, once sentences are represented as set of their predicate-argument tuples \( \psi \), unpaired tuples in a sentence pair are identified. This can be learnt by training a decision tree classifier on manually labelled sample of sentence pairs (around 200). In MSR paraphrase corpus, following sentence-pair types are relevant for first sub-task:

- **SP**: Sentence pair where each sentence has only one predicate-argument tuple
- **SP**: Sentence pair that is paraphrasing and has only one paired tuple (loosely or semantically) and only one unpaired tuple

Predicate-argument tuple pairs belonging to \( SP \) and \( SP \) sentence pairs make data-set for first task, where predicate-argument tuple pair is labelled similar if sentence pair is paraphrasing and dissimilar otherwise. Predicate-argument tuple-tuple similarity matrix is used as features for training tuple semantic similarity classifier \( \bar{\xi} \).

Using \( \bar{\xi} \), paraphrase training data set is tested for semantically paired and loosely paired tuples. For learning
dissimilarity significance in sentence pair paraphrasing, this paper follows Qiu’s approach [11] except that authors defined dissimilarity in terms of unpaired tuples only while our work defines dissimilarity in terms of unpaired and loosely paired tuples. Following sentence-pair types are relevant for creation of dissimilarity significance classification data set:

**UP**: Sentence pair that is non-paraphrasing; and that has only one unpaired or only one loosely paired tuple and all paired tuples are semantically paired.

**UP**: Sentence pairs that is paraphrasing; and that has at least one semantically paired tuple.

Unpaired tuple or loosely paired tuple in sentence pair belonging to **UP** is significant, as despite of all paired tuples being similar the sentence pair is non-paraphrasing. Similarly, all unpaired tuple or loosely paired tuples in sentence pair of type **UP** are insignificant as despite of its presence, the sentence pair is paraphrasing. Qiu et al [11] used n-gram (n = 4) syntactic path between predicate of unpaired tuple and paired tuple with the closest shared ancestor. Apart from four-gram syntactic path features, other features presumed to be of significance for the task are WordNet [18] verb category (after verb sense disambiguation as mentioned in VSM based representation scheme above), number of children tuple’s predicate has in its sentence’s transformed parse tree, and whether predicate has any arguments other than subject (arg0).

3.3 **Sentence Paraphrase Recognition Testing**

A sentence pair is paraphrasing if it has at least one semantically paired tuples and all its dissimilarities (loosely paired and unpaired tuples) are insignificant in paraphrasing or meaning.

Methodology can be summarised as follows:

**Input:**

- Sentence Pair: \( S_1, S_2 \)
- Labelled training Data-set: \( D_r \) with \( N \) sentence pairs \( \{ S_{1i}, S_{2i}, p_i \} \) where \( p_i = 1 \) if \( i^{th} \) sentence pair is paraphrases and 0 otherwise

**Output:** Predicted paraphrase status \( P \) (0/1) for SS

**TRAINING BEGIN**

**Step 1:** Represent each sentence in \( D_r \) as set of its predicate-argument tuples using \( \psi \)

**Step 2:** For each sentence pair in \( D_r \), identify unpaired tuples and paired (loosely/semantically paired) tuples using heuristic \( K \)

**Step 3:** Select paired tuples of \( SP_1 \) and \( SP_2 \) type sentence pairs to form a training data-set \( D_r^p \) for training tuple paraphrase classifier \( \xi_p \)

**Step 4:** Create tuple-tuple similarity matrix as feature set to learn \( \xi_p \)

**Step 5:** Label each tuple pair in \( D_r^p \) as paraphrasing if corresponding sentence pair is paraphrasing and non-paraphrasing otherwise

**Step 6:** Train \( \xi_p \)

**Step 7:** For each sentence pair in \( D_r \), identify paraphrasing tuples and loosely paired tuples using \( \xi_p \)

**Step 8:** Select unpaired and loosely paired tuples of \( UP_1 \) and \( UP_2 \) type sentence pairs to form a training data-set \( D_r^p \) for training dissimilarity significance classifier \( \xi_d \)

**Step 9:** Create following feature-set for learning \( \xi_d \):

- n-gram features of shortest parse tree path of tuple’s predicate from predicate of any paired tuples of the sentence.
- WordNet verb category

**Step 10:** Label each tuple pair in \( D_r^p \) as significant if corresponding sentence pair is of type \( UP_1 \) and non-paraphrasing if it is of type \( UP_2 \)

**Step 11:** Train \( \xi_d \)

**TRAINING END**

4 **Conclusion and Future Directions**

Semantic role labels or predicate-argument tuples are the smallest grammatical units using which a sentence meaning can be appropriately captured. Qiu et al [11] proposed sentence paraphrase recognition methodology using predicate-argument tuples as the basic unit of information. However, Qiu et al [11] approach lacks in efficient SRL representation methodology and relies on thesaurus based heuristic to identify paraphrasing tuples. This paper proposed two improvisations on Qiu et al [11] approach. First, two tuple representation schemes are proposed – VSM based and RAE based representations. The two vector based representation schemes are asserted to deliver faster and accurate SRL phrase comparison. Second, the paper introduced concept of loosely paired tuples in order to formalize tuple paraphrase recognition problem as a separate classification task. For comparing two tuples, use of tuple-tuple similarity metric is suggested as it efficiently captures SRL alignment corresponding to polysemous verb frames. Paper also proposes a verb sense disambiguation algorithm which has been validated manually on 200 sentence pairs from MSR paraphrase corpus.

The paper proves to be a blueprint for SRL based sentence paraphrase recognition. Further, one should verify which representation scheme best captures the paraphrasing features of a sentence pair. Also, appropriate handling of null SRL
comparisons in tuple-tuple similarity matrix is required for an efficient paraphrase classifier. SRL is one of the most basic unit of information with which meaning of a text can be comprehended. Formalization proposed for tuple representation and for tuple paraphrase recognition are useful for any SRL based NLP task in general.

5 References


