# INVESTIGATING PARAPHRASING ALGORITHMS WITH APPLICATION TO SPOKEN DIALOGUE SYSTEMS

By
Arodami Chorianopoulou

DIPLOMA THESIS
TECHNICAL UNIVERSITY OF CRETE
CHANIA, GREECE
OCTOBER 2013

Supervisor:

Assist. Prof. Polychronis Koutsakis

Readers:

Assoc. Prof. Alexandros Potamianos

Prof. Evripidis Petrakis

*To Manolis and Maria*

# Table of Contents

# List of Tables

# List of Figures

# Abstract

The task of paraphrasing is an important part of natural language processing (NLP) and is being employed in several NLP applications. A paraphrase is an alternative surface form in the same language expressing the same semantic content as the original form. In this thesis, we investigate paraphrase recognition and generation via a composition of Finite State Machines (FSMs) combined with english resources. Additionally, we propose several string similarity measures for paraphrase recognition of sentences.

An FSM defines a language by accepting a string of input tokens in the language and rejecting those that are not included in the language. In our case, the tokens are the words of the language. Our motivation was to create an alignment between sentences by using FSMs. This linking between two sentences allows as to identify whether or not they are paraphrases. Then, by adding a language model we manage to generate paraphrases.

To test our idea, we experimented on domain-specific paraphrasing, and more specifically, in the travel domain field, by employing paraphrase recognition and generation on prompts. For this purpose, we have used two different types of corpora. Our first corpus contains web documents harvested from the web while the second was created from the CrowdFlower crowdsourcing service.

To evaluate the performance of the method we used the datasets from the SemEval workshop of 2012, in order to estimate the semantic equivalence between sentences. The comparison of the results at the evaluation stage with scores from string-based similarity metrics, shows that our implementation achieves encouraging performance.

# Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisor, Prof. Alexandros Potamianos for his valuable guidance and advice. His willingness to motivate me contributed tremendously to my thesis.

I also wish to thank Dr. Elias Iosif, who was abundantly helpful and offered invaluable assistance and support.

Deepest gratitude is also due to the members of the supervisory committee, Prof. Polychronis Koutsakis, for his support during the last months and Prof. Euripidis Petrakis.

Finally, an honorable mention goes to my family, my parents and my sister, Elisabeth, and my friends, Alexandros, Angeliki, Antonela, Vaggelis, Voula, Ioannis L., Ioannis T., Irini, Kwstas, Sotiris, Stella, for their understanding and support to me while I worked on this thesis. Without their helps, I would have faced many difficulties in completing my work.

# Chapter 1

# Introduction

The concept of paraphrasing is most generally defined on the basis of the principle of semantic equivalence: A paraphrase is an alternative surface form in the same language expressing the same semantic content as the original form. Paraphrasing may occur in several levels.

Paraphrasing methods recognize, generate or extract phrases, sentences or longer natural language expressions that convey almost the same information. It is a bidirectional method that can be used in natural language processing applications, including question answering, summarization, text generation, machine translation. On the other hand, textual entailment methods recognize, generate or extract pairs of natural language expressions, such that someone who decides that the first sentence is true would most likely infer that the second sentence is also true. Paraphrasing can be seen as a bidirectional textual entailment.

## 1.1 Applications

Paraphrasing has in many cases been developed for question answering systems. In such a system, a question may be phrased differently so that the answer may not correspond to the expected one. Such variations can also improve system's performance [2][23][28][59]. A similar application, which also uses question paraphrasing is to obtain the most frequent questions (FAQs).

In text summarization [2][47][31], an important stage is to identify the most important sentences of the text to be summarized. The goal is to avoid sentences that convey almost the same meaning with others that have already been selected.

Another application is sentence compression [2][37][50][21][25], in which the constraint is that the resulting sentence must be shorter than the original one but still grammatically correct. In most cases sentence compression drops the less important information of the original sentence.

Information extraction systems [2][26][52] locate text snippets that are connected to particular types of events. In this case, paraphrasing can be used to generate additional semantically equivalent extraction patterns.

Paraphrasing has improved machine translation [2][38] measures and processes at

the evaluation stage. Moreover, paraphrasing has enriched translation systems and improved their performance, as it allows them to cope with words or phrases that have not been contained at the training corpora.

In natural language generation [2][8], paraphrasing can be used to avoid repeating the same phrasings or to produce alternative expressions. Among those applications, paraphrasing methods can be employed to simplify texts, for example by replacing specialized terms with non-expert expressions.

## 1.2 Classification of the methods

To provide a clear view of the different goals and assumptions of the methods that have been proposed over the years, we classify paraphrasing in many dimensions [2][55]. First, we classify the methods based on whether they perform detection, recognition, generation or extraction. This distinction, in many cases, may not be clear. The second classification refers to where the paraphrasing has been employed. There are three levels, lexical, phrasal and sentential. Finally, we can classify the methods depending on the type of corpora they use. That can be a single monolingual corpus, monolingual comparable corpora, monolingual parallel corpora and bilingual parallel corpora.

Paraphrase detection and recognition refer to the problem of assigning a measurement to the semantic similarity of two phrases. The main input to a paraphrase recognizer (detector) is a pair of expressions. The output is a judgement indicating whether or not the input pair of sentences are paraphrases. The difference between detection and recognition relies to the output measurement they provide. In the first case, the annotation is either 0 or 1 for "not paraphrase" or "paraphrase" input pair respectively, while in recognition the output is a continuous value ranging from 0 to 1. On the other hand, the input to a paraphrase generator is a single language expression. The output is a set of candidate paraphrases of the input phrase and it must be as large as possible. Finally, in the case of a paraphrase extractor, the main input is a corpus while the output consists of pairs of paraphrases. The goal is to produce as many output pairs as possible.As it has already been mentioned, the boundaries between the above classifications may not always be clear. For example, in many cases a paraphrase generator may use a recognizer to filter out candidate paraphrases.

At the lexical level, lexical items having the same meaning are mostly referred as lexical paraphrases or synonyms. However, there are forms of lexical paraphrasing that can not be described as synonyms, such as hyperonymy where one of the words is either more general or more specific than the other. The term phrasal paraphrase refers to phrases which have the same semantic content. The phrasal fragments may take the form of syntactic phrases or they may be patterns with linked variables.

Finally, paraphrasing can be seen in sentances, i.e. sentential paraphrasing. To create sentential paraphrases, one can simply substitute words or phrases with semantic equivalents. However, this technique may not be able to generate more complex ones.

Regarding our final distinction, we have notified 4 different types of corpora. In linguistics a corpus is a large and structured set of texts. Nowadays and in most cases corpora are stored electronically. They are used for linguistics process and analysis. A single monolingual corpus contains texts of only one language. Most of the existing

corpora are in English, however there are a few in European, Middle Eastern and East Asian languages. Monolingual comparable corpora contain texts or in many cases articles that refer to same events or same topics. Monolingual corpora that have been specially formatted for side-by-side comparison are called aligned parallel corpora. For example, multiple translations of the same novel are used as parallel corpora. A corpus that contains texts in multiple languages is called multilingual. In many NLP applications, corpora containing 2 different languages are used, in which case they are called bilingual. A method using bilingual corpora would translate a text or a sentence into one language (called pivot language) and then back to the original one.

## 1.3 Purpose of the Thesis

In this thesis we present paraphrasing methods and semantic textual similarity by using a variety of string similarity measures and an alignment based on Finite State Machines (FSMs). The rest of the thesis is organized as follows.

*Chapter 2* contains related work for paraphrasing methods and Semantic Textual Similarity. We present the main ideas, algorithms and evaluation procedure for our subject of interest.

*Chapter 3* presents our approach on paraphrase detection, recognition and generation as well as on semantic textual similarity. We describe the string similarity measures and the FSM-based alignment.

*Chapter 4* presents our experiments and the evaluation results for our tasks. Additionally, we describe the different kind of corpora that we have tested our approach on.

Finally *Chapter 5* presents our conclusions and the opportunities for future work that can be derived from our study.

# Chapter 2

# Related Work

Paraphrases are alternative ways to convey the same information. Recognizing, extracting or generating semantically equivalent phrases is of significant importance in many natural language applications, hence in the past paraphrases have come under the scrunity of many research communities. What distinguishes these applications is not only the volume of research devoted to them but the fact that for each one there is typically a well-defined problem setting, a standard metric for evaluating the task, standard corpora on which the task can be evaluated, and competitions devoted to the specific task.

In this section, first we define the problem and then we consider in turn recognition, generation and extraction methods for paraphrasing. In each of the three categories we explain the ideas and the resources used in past approaches. Moreover, we present the idea of Semantic Textual Similarity through the SemEval workshop.

## 2.1 Problem Statement

In a paraphrasing system the need to estimate the semantic equivalence between two sentences is crucial. Therefore, it is important to come up with a process which will be able to recognize, extract or generate paraphrases, yielding as a result sentences (or phrases or patterns) that will maintain the semantic notion of the original sentence. As semantic notion we understand the meaning of the sentence which relies on the relation between words or phrases. The distiction between the three methods may not always be clear, for instance paraphrase generation may be combined with a recognition system.

In a paraphrase recognition system, one of the most important issues, is to determine the semantic equivalence between two sentences, which defines how similar the sentences are. More specifically, a system should return a score for a pair of sentences. As the score increases to one, the sentences tend to have similar meaning. At that point, the system can capture the semantic similarity, which can be seen as a different task. In order to complete the recognition method we have to determine a threshold, which will classify the pair of sentences as paraphrases or not. Applying the threshold effectively is an issue of great importance, however in our work this is considered as a machine learning problem.

The information extracted from the recognition task is widely used for generation and extraction systems. In a generation system, the goal is to create as many sentences as possible that are semantically equivalent to an original one. The simpliest approach for such systems is substituting the words of the original sentence with synonyms, which can be obtained from several english resources. Furthermore, paraphrases could be generated using more complicated ways. For instance, by building a set of paraphrasing rules, using statistical machine translation, or transforming the sentence into a representation (graph, FSM, etc.) and employing a system which builds a generator. In the last approach, the system is able to understant the sentence and present the meaning in another way. Obviously, the design of such a system depends on the way the sentences are aligned. A recognition system can be employed to filter out the generated paraphrases.

Furthermore, the last of the paraphrasing methods intends to extract as many as possible paraphrases from a corpus. For this purpose, systems have been developed using the distributional hypothesis, which implies that words appearing in the same context tend to have similar meaning. Moreover, bootstrapping methods have been used. The obvious problem, is the one of alignment. However, extraction methods are not addressed in this thesis.

One can easily see that in scenarios like the ones described above, the estimation of the semantic similarity and creating an alignment between two sentences are of great importance. The goal of this thesis is to provide an effective system which copes with paraphrase recognition and generation, as well as semantic textual similarity.

## 2.2 Semantic Textual Similarity

Semantic textual Similarity (STS) measures the degree of semantic equivalence between two texts. STS is related to both Textual Entailment (TE) and Paraphrase (PARA). STS is more directly applicable in a number of NLP tasks than TE and PARA, such tasks as Machine Translation and evaluation, Summarization, Machine Reading, Deep Question Answering, etc. Additionally, STS differs from PARA in that, rather than being a binary yes/no decision, STS incorporates the notion of graded semantic similarity.

### 2.2.1 Similarity Measures

Plenty of word similarity measures can be used in order to estimate the semantic equivalence between two sentences. These can be categorized as follows:

1. Corpus-based
   Measures that try to identify the similarity of words using information exclusively from large corpora. Specifically, these are features which capture the context of the sentence by creating vectors. There are several features, like ESA, LSA, PMI, etc. which create vectors based on the context [5][1] [19].

2. Knowledge-based
   Measures that rely on a semantic network of words (e.g. WordNet, Wikipedia

etc.) [11][17].
For example, WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of synonyms (synsets), each expressing a distinct concept. Synsets are linked by means of conceptual-semantic and lexical relations [24].

To estimate the sentence similarity, one can use lexical, syntactic or semantic features. In the first case, there is only comparison of the surface form of the word, while syntactic or semantic relations are used for the other cases respectively. Additionally, we should mention the alignment methods used, which vary among the systems.

- **Lexical Level**
  In computer science, lexical analysis is the process of converting a sequence of characters into a sequence of tokens. At the lexical level of sentential similarity, several similarity measures could be explored, such as: word or character n-grams, edit distance, least common subsequence (LCS), least common substring (LCSstring), resemblance coeeficients (e.g. Jaccard, Dice), etc. Moreover, lexical semantic features may derive from Machine Translation (e.g. BLEU) [5][35].

- **Syntactic Level**
  At the syntactic level, similarities are based on the dependency relations among the words of a sentence. The dependency relation views the (finite) verb as the structural center of all the clause structure. All other syntactic units are either directly or indirectly dependent on the verb. Syntactic dependencies can be represented as structures, most commonly tree structures. A structure is determined by the relation between a word (a head) and its dependents [66][54].

- **Semantic Level**
  At the semantic level, in order to estimate the similarity, we can either use lexical entailment or classical semantic relations. In the second case, we refer to such relations, as synonyms, hypernyms etc.

### 2.2.2 Alignment

The alignment between two sentences can be employed in several ways. Besides using similarity features, as already described, there are a variety of tools, which can create an alignment. These include FSMs, graphs, lattices etc.

***Universal Networking Language (UNL)*** is a declarative formal language specifically designed to represent semantic data extracted from natural language texts. It can be used as a pivot language in interlingual machine translation systems or as a knowledge representation language in information retrieval applications [64].

***Finite State Machines*** It is conceived as an abstract machine that can be in one of a finite number of states. The machine is in only one state at a time; the state it

is in at any given time is called the current state. It can change from one state to another when initiated by a triggering event or condition; this is called a transition. A particular FSM is defined by a list of its states, and the triggering condition for each transition [72][7].

An important distinction among the FSMs is based on the number of tapes. More specifically, an FSM with one tape is called "acceptor" and its functionality is to accept a specific language, while an FSM with two tapes is called "transducer". A transducer has the ability to convert an input language to an output one according to the transitions allowed.

There are several FSM-toolkits, which support a variety of FSM functions. The most common of these are: compile, print and draw (for compilation and display), concatenation, union, intersection, epsilon removal and composition (for construction and combination), minimization and determinization (for minimization and equivalence) and best path for fsm searching. The definition of each fsm function is described in Appendix A: The FSM Toolkit.

***Graphs*** Graphs can be used to represent natural language text according to the contextual relations of words in higher-level language units (e.g. sentences, definitions or documents). In these graphs, words and/or higher-level language units are represented with nodes, and edges are added between them according to their textual context, syntactic relatedness, etc [3] [22].

### 2.2.3 Machine Learning

Machine learning is about the construction and study of systems that can learn from data. There are two types: supervised and unsupervised, and both can be applied to Semantic Textual Similarity.

1. **Supervised Machine Learning** is the machine learning task ([20][10]) of inferring a function from labeled training data. The training data consists of training examples, accompanied with a desired output value. Such an algorithm analyzes the training data and produces an inferred function, which is called either a classifier or a regression function. The inferred function should predict the correct output value for any valid input object.
   Some commonly used supervised algorithms are:

   ***Linear Regression*** A linear classifier computes the correlation by making a classification decision based on the value of a linear combination of the features. An object's features are typically presented to the machine in a vector called feature vector. If the input feature vector to the classifier is a real vector $\vec{x}$, then the output score is:

   $$y = f(\vec{w} \cdot \vec{x}) = f(\sum_j w_j x_j) \tag{2.2.1}$$

   where $\vec{w}$ is a real vector of weights and $f$ is a function that converts the scalar product of the two vectors into the desired output. The weight vector $\vec{w}$ is learned

from a set of labeled training samples.

***Bagging classifier*** A bagging classifier is a "bootstrap" method because it trains each classifier on a random redistribution of the training set. Each classifier's training set is generated by randomly drawing with replacement N'¡ N examples, where N is the size of the original training set. As a result many of the original examples may be repeated in the resulting training set while others may be left out. The N' models are fitted using the above N' examples and combined by averaging the output (for regression) or voting (for classification).

***M5P classifier*** An M5P classifier implements base routines for generating M5 Model trees and rules. Model trees are a type of decision tree with linear regression functions at the leaves and are able to predict continuous numeric values. They can be applied to classification problems by employing a standard method of transforming a classification problem into a problem of function approximation.

***Naive Bayes classifier*** A naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. The naive Bayes classifier combines the a probabilistic model with a decision rule. One common rule is to decide in favour of the hypothesis that is more probable; this is known as the MAP decision rule. The corresponding classifier is the function defined as follows:

$$classify(f_1, ..., f_n) = \arg\max_c p(C = c) \prod_{i=1}^{n} p(F_i = f_i | C = c) \tag{2.2.2}$$

***SVM classifier*** Support Vector Machines are supervised learning models with associated learning algorithms that analyze data and recognize patterns. The basic SVM takes a set of input data and predicts, for each given input, which of two possible classes forms the output, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, divided in categories. New examples are then mapped into that same space and predicted to belong to one of the categories, based on their distance from them.

Given some training data $D$, a set of $n$ points of the form

$$D = \{(x_i, y_j) | x_i \in R_p, y_j \in \{-1, 1\}\}_{i=1}^{n} \tag{2.2.3}$$

where the $y_i$ is either 1 or -1, indicating the class to which the point $x_i$ belongs.

2. **Unsupervised Machine Learning** It refers to the problem of trying to find hidden structures in unlabeled data. Since the examples given to the learner are unlabeled, there is no error or reward signal to evaluate a potential solution.

### 2.2.4 Evaluation

Experimenting with semantic textual similarity requires datasets containing both positive and negative input pairs. Given two sentences, s1 and s2, an STS system will return a score. For the evaluation procedure of the output, Pearson and Spearman correlation coeeficients are widely used.

***Pearson correlation*** is a measure of the linear correlation (dependence) between two variables X and Y, giving a value between +1 and -1 inclusive, where 1 is total positive correlation, 0 is no correlation, and -1 is negative correlation. It is widely used in the sciences as a measure of the degree of linear dependence between two variables. Pearson's correlation coefficient between two variables is defined as the covariance of the two variables divided by the product of their standard deviations.

***Spearman correlation*** is a nonparametric measure of statistical dependence between two variables. It assesses how well the relationship between two variables can be described using a monotonic function. If there are no repeated data values, a perfect Spearman correlation of +1 or -1 occurs when each of the variables is a perfect monotone function of the other.Spearman's coefficient, like any correlation calculation, is appropriate for both continuous and discrete variables, including ordinal variables.

## 2.3 Paraphrase Detection & Recognition

Paraphrase detection decides whether or not two given phrases (or sentences) are paraphrases. Methods at different levels of the input sentences have been employed. For example, they may operate on the semantic or syntactic representation or the combination of different levels.

### 2.3.1 Logic-based

Logic-based approaches are based on mapping the language expressions to logical meaning representations, and then rely on logical entailment checks, possibly by invoking theorem provers [2][12][69]. English resources can provide sense knowledge, axioms or semantic frames. Specifically, WordNet and FrameNet can be used, or online encyclopedias for background knowledge by extracting particular types of information from their articles.

### 2.3.2 Vector Space Models of Semantics

In such approaches each word of the input phrase is mapped to a vector which shows how strongly the word cooccurs with particular other words in corpora [2] [41], possibly also taking into account syntactic information. A compositional vector-based can then be used to combine the vectors of single words; in the simplest case, the vector of each

expression could be the sum or product of the vectors of its words. Paraphrases are detected by measuring the distance between the vectors using, for example, the cosine similarity.

### 2.3.3 Surface String Similarity

Several paraphrase recognition methods operate on the input surface string without computing more syntactic or semantic representations. However, it is possible, for a pre-processing stage to be applied, such as part-of-speech (POS) tagging, lemmatization or named-entity recognition. For example, some features that are usually computed are the number of common words or combinations of several string similarity measures [2] [44] including measures originating from machine translation (e.g. BLEU), the string edit distance [40].

### 2.3.4 Syntactic Similarity

In paraphrasing, another common approach is working at the syntax level. Specifically, the dependency grammar parsers are quite popular. Their output is a graph (usually a tree) whose nodes are words of the sentence and whose edges correspond to the syntactic dependencies between words.

To estimate the similarity someone could count the common edges of the dependency trees of the input pair or use other tree similarity measures (e.g. tree edit distance). A large similarity score implies that the input sentences are paraphrases. Tree edit distance [2][62] [68][74] computes the sequence of operator applications (e.g. add, replace, remove an edge or a node) with the minimum cost that turns one tree into the other. To obtain more accurate predictions, it is important to assign appropriate costs to the operators during a training stage.

### 2.3.5 Similarity Measures Operating on Symbolic Meaning Representations

Graphs with edges corresponding to semantic relations instead of syntactic dependencies may also recognize paraphrases [2][27]. Relations of this kind may be identified by applying semantic role labeling methods [49] to the input pair of sentences. Several resources, such as FrameNet, PropBank, Wordnet etc. provide semantically related phrases or words. For example, using FrameNet's frames has the advantage that semantically related expressions may invoke the same frame or interconnected frames, making similarities and implications easier to capture. WordNet automatically constructed collections of near synonyms [42][53][16].

By treating semantically similar words as identical, paraphrase recognizers may be able to cope with paraphrases that have very similar meanings, but few or no common words.

### 2.3.6 Employ Machine Learning

Using machine learning, multiple similarity measures, possibly computed at different levels, can be combined. Each sentence of the input pair is represented as a vector which contains the scores of every measure that has been applied to the pair and possibly other features. For example, many systems also include features that check for polarity differences across the two input expressions [2][27] [34][70]. A supervised machine learning algorithm trains a classifier with manually classified vectors which correspond to pairs of sentences (training input pairs). Once trained, the classifier can classify inseen pairs as correct or incorrect paraphrases by simply examining their features. In such a recognizer, a preprocessing stage is usually applied, which contais POS tagging, parsing or normalization.

Instead of mapping each pair to a feature vector that contains mostly scores measuring the similarity between the two input sentences, it is possible to use vectors that encode directly parts of their syntactic or semantic representation.

### 2.3.7 Decoding

Decoding is a recognition approach which uses a sequence of rules, that may have been produced by extraction mechanisms, often in addition to synonyms and hypernyms-hyponyms, in order to turn one of the input sentences to the other. If there is such a sequence then the pair of sentences are paraphrases depending on the rules used. Each rule is associated with a confidence score (possibly learnt from a training dataset) so that the degree to which the rule preserves the original meaning is reflected. Then we can search for the sequence of transformations with the maximum score (or minimum cost) [2][29]. The context where rules are applied may also be used as there may exist words with multiple senses.

Resources like WordNet and extraction methods provide such rules. When operating at the level of semantic representations, the sequence sought is in effect a proof that the two input expressions are paraphrases and it may be obtained by exploiting theorem provers [6].

### 2.3.8 Evaluation

Experimenting with paraphrase recognizers requires datasets containing both positive and negative input pairs. When using discriminative classifiers (e.g. SVMs) the negative training pairs must ideally be near misses, otherwise they may be of little use [2][61][71].

The most widely used benchmark dataset for paraphrase recognition is the Microsoft Research (MSR) Paraphrase Corpus. It contains 5,801 pairs of sentences obtained from clusters of online news articles reffering to the same events. The pairs were initially filtered by heuristics and then filtered by an SVM-based paraphrase recognizer, trained on separate manually classified pairs, which was biased to overidentify paraphrases. Finally, human judges annotated the remaining sentence pairs as paraphrase or not. Approximately, 67% of the 5,801 pairs were judged as paraphrases. The dataset is divided in two non-overlapping parts, for training (70% of all parts) and testing (30%).

Table 2.1 lists all the published results of paraphrase recognition experiments on the MSR corpus. Two baseline systems are included: $BASE_1$ classifies all pairs as paraphrases; $BASE_2$ classifies two sentences as paraphrases when their surface word edit distance is below a threshold, tuned on the training part of the corpus.

| method | accuracy (%) | precision (%) | recall (%) | F-measure (%) |
|---|---|---|---|---|
| Corley & Mihalcea (2005) | 71.5 | 72.3 | 92.5 | 81.2 |
| Das & Smith (2009) | 76.1 | 79.6 | 86.1 | 82.9 |
| Finch et al. (2005) | 75.0 | 76.6 | 89.8 | 82.7 |
| Malakasiotis (2009) | 76.2 | 79.4 | 86.8 | 82.9 |
| Qiu et al. (2006) | 72.0 | 72.5 | 93.4 | 81.6 |
| Wan et al. (2006) | 75.6 | 77.0 | 90.0 | 83.0 |
| Zhang & Patrick (2005) | 71.9 | 74.3 | 88.2 | 80.7 |
| $BASE_1$ | 66.5 | 66.5 | 100.0 | 79.9 |
| $BASE_2$ | 69.0 | 72.4 | 86.3 | 78.8 |

Table 2.1: Paraphrase recognition results on the MSR corpus

Four commonly used evaluation measures are used: precision 2.3.1, recall 2.3.2, accuracy 2.3.3 and F-measure 2.3.4 with equal weight on precision and recall. These measures are defined below.

$$PR = \frac{TP}{TP + FP} \qquad (2.3.1) \qquad RC = \frac{TP}{TP + FN} \qquad (2.3.2)$$

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \qquad (2.3.3) \qquad FM = \frac{2 * PR * RC}{PR + RC} \qquad (2.3.4)$$

where :

- TP (true positives) are the numbers of pairs that have been correctly classified as positives (paraphrases).
- FP (false positives) are the numbers of pairs that have been incorrectly classified as positives.
- TN (true negatives) are the number of pairs that have been correctly classified as negatives (not paraphrases).
- FN (false negatives) are the number of pairs that have been incorrectly classified as negatives.

From the results of Table 1 we can see that all systems have better recall than precision, which implies that they tend to over-classify pairs as paraphrases, possibly because the sentences of each pair have at least some common words. The high F-measure of $BASE_1$ is largely due to its perfect recall; its precision is significantly

lower, compared to the other systems. $BASE_2$, which uses only string edit distance, is a competitive baseline for this corpus.

## 2.4 Paraphrase Generation

Unlike paraphrase recognizers, which can be embedded in paraphrase generators to filter out erroneous generated paraphrases, generators are modules that produce new phrases, patterns or sentences that are semantically equivalent. Paraphrase generators are given as input a single sentence and they produce as many output candidate paraphrases as possible. There are two main approaches for paraphrase generators.

### 2.4.1 Inspired by Statistical Machine Translation

Many generation methods borrow ideas from statisical machine translation (SMT). SMT methods rely on very large bilingual or multilingual corpora [2] [55]. Let's assume a sentence F, whose words are $f_1, f_2, ..., f_{|F|}$ and N one candidate translation, whose words are $a_1, a_2, ..., a_{|N|}$ . The best translation, denoted $N\star$, is the N with the maximum probability of being a translation of F.

$$N\star = \arg\max P(N|F) = \arg\max \frac{P(N)P(F|N)}{P(F)} = \arg\max P(N)P(F|N) \quad (2.4.1)$$

Since F is fixed, the denominator P(F) above is fixed and can be ignored when searching for $N\star$. P(N) is called the language model and P(F—N) the translation model.

Assuming that the probability of encountering word $a_i$ depends only on the preceding n-1 words, P(N) becomes :

$$P(N) = P(a_1) \cdot P(a_2|a_3) \cdot P(a_3|a_1, a_2) \cdot P(a_4|a_2, a_3) \cdots P(a_{|N|}|a_{|N|-2}, a_{|N|-1}) \quad (2.4.2)$$

A language model also includes smoothing mechanisms, to deal with n-grams that are rare or not present in the corpus, which would lead to P(N)=0.

To create the corpus, methods like those employed to create the MSR corpus have been used. The sentence pairs are then word aligned as in machine translation, and the resulting alignments are used to create a table of phrase pairs as in phrased-based SMT systems [39]. The table of phrase pairs may include synonyms obtained from WordNet or similar resources or pairs of paraphrases discovered by paraphrase extraction methods. Phrase pairs that occur frequently in the aligned sentences may be assigned higher probabilities [58]. Their decoder first constructs a lattice that represents all the possible paraphrases of the input sentence that can be produced by replacing phrases by their counterparts in the phrase table. Unlike machine translation, not all of the words or phrases need to be replaced.

Paraphrases can also be generated by using pairs of machine translation systems to translate the input expression to a new language, often called a pivot language, and then back to the original language. The resulting expression is often different from the input one, especially when the two translation systems employ different methods [23].

An advantage of this approach is that the machine translation systems can be treated as black boxes, and they can be trained on readily available parallel corpora of different languages. A disadvantage is that translation errors from both directions may lead to poor paraphrases.

An approach similar to syntactic transfer in machine translation may also be adopted [51]. In that case, the input expression is first parsed. The resulting syntactic representation is then modified in ways that preserve the original meaning. New language expressions are then generated from the new syntactic representations. Parsing, however, the input expression may introduce errors, and producing a correct meaning representation of the input may be far from trivial.

### 2.4.2 Bootstrapping

When the input and output expressions are slotted templates, it is possible to apply bootstrapping to a large monolingual corpus, instead of using machine translation methods. Initially, we retrieve from the corpus sentences that contain seed pairs which match to the template we wish to generate. Simple filtering techniques may be used, because some of the generated candidate templates may not be paraphrases of the original one. Having obtained new templates, we can search the corpus for new sentences that match them. From the new sentences, more seed values can be extracted. More iterations may be used to generate more templates and more seeds, until no more templates and seeds can be discovered or a maximum number of iterations is reached.

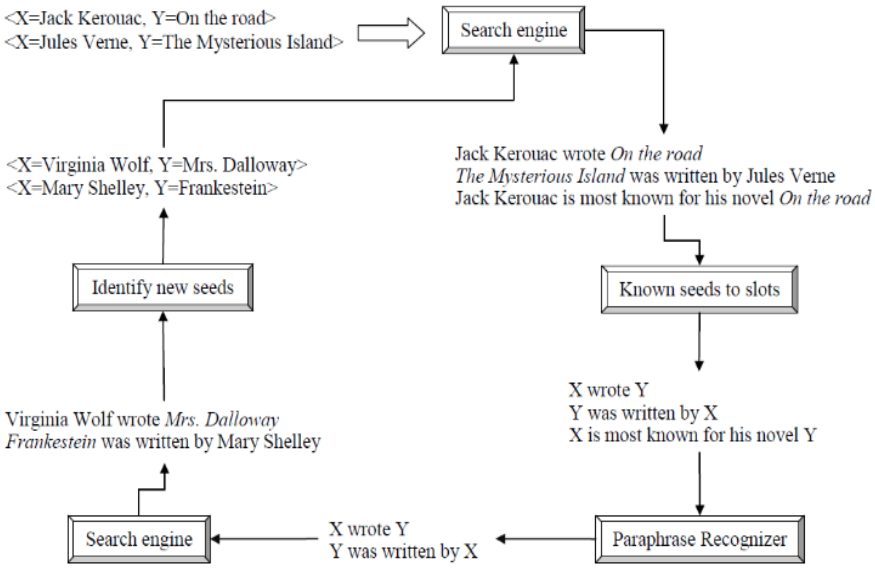



Figure 2.1: Generating paraphrases of "X wrote Y" by bootstrapping ([2]).

If slot values can be recognized reliably, we can obtain the initial seed slot values automatically by retrieving directly sentences that match the original templates. Moreover, we can obtain the initial templates automatically, by identifying sentences of interest, identifying slot values in the sentences and using the contexts of the slot values as initial templates. TEASE [67] is a well-known bootstrapping method of this kind.

Similar bootstrapping methods have been used to generate information extraction patterns [2][60][73]. Some of those methods, however, require corpora annotated with instances of particular types of events to be extracted [32][65][18].

### 2.4.3 Evaluation

In most paraphrase applications, it is desirable not only to produce correct outputs, but also to produce as many correct outputs as possible. The two goals correspond to high precision and recall respectively [2].

For a particular input $s_i$, the precision $p_i$ and recall $r_i$ of a generator can be defined as follows. $TP_i$ is the number of correct outputs for input $s_i$, $FP_i$ is the number of wrong outputs for $s_i$ and $FN_i$ is the number of outputs for $s_i$ that have incorrectly not been generated.

$$p_i = \frac{TP_i}{TP_i + FP_i} \qquad (2.4.3)$$

$$r_i = \frac{TP_i}{TP_i + FN_i} \qquad (2.4.4)$$

Instead of reporting recall, it is common to report (along with precision) the average number of outputs, sometimes called yield, defined below, where we assume that there are n test inputs. A better option is to calculate yield at different precision levels, since there is usually a tradeoff between the two figures, which is controlled by parameter tuning.

$$yield = \frac{1}{n}\sum_{i=1}^{n}(TP_i + FP_i) \qquad (2.4.5)$$

Unlike recognition, there are no widely adopted benchmark datasets for paraphrase generation, and comparing results obtained on different datasets is not always meaningful. The lack of generation benchmarks is probably due to the fact that although it is possible to assemble a large collection of input expressions, it is practically impossible to specify in advance all the numerous correct outputs a generator may produce.

## 2.5 Paraphrase Extraction

Unlike recognition and generation methods , extraction methods are not given particular input language expressions. They typically process large corpora to extract pairs of language expressions that constitute paraphrases. The generated pairs are stored to be used subsequently by recognizers and generators or other applications [2].

### 2.5.1 Distributional Hypothesis

A possible paraphrase extraction approach is to store all the word n-grams that occur in a large monolingual corpus (e.g. for $n \leq 5$), along with their left and right contexts, and consider as paraphrases n-grams that occur frequently in similar contexts. Vector similarity measures, for example cosine similarity or Lin's measure [42], can then be employed to identify n-grams that occur in similar contexts by comparing their vectors. This approach has been shown to be viable with very large monolingual corpora.

Paraphrasing approaches of this kind are based on Harris's Distributional Hypothesis [30], which states that words in similar contexts tend to have similar meanings.

Lin and Pantel's (2001) [43] well-known extraction method, called DIRT, is also based on the extended Distributional Hypothesis, but it operates at the syntax level. DIRT first applies a dependency grammar parser to a monolingual corpus. Dependency paths are then extracted from the dependency trees of the corpus. Once the paths have been extracted, it looks for pair of paths that occur frequently with the same slot fillers. A measure based on mutual information [48] is used to detect paths with common fillers.

Bhagat et al. (2007) [9] developed a method, called LEDIR, to classify the template pairs $< P_1, P_2 >$ that DIRT and similar methods produce into three classes: (i) paraphrases, (ii) $P_1$ textually entails $P_2$ and not the reverse, or (iii) $P_2$ textually entails $P_1$ and not the reverse; with the addition of LEDIR, DIRT becomes a method that extracts separately pairs of paraphrase templates and pairs of directional textual entailment templates. Ibrahim et al.'s (2003) [33] method is similar to DIRT, but it assumes that a monolingual parallel corpus is availiable (e.g. multiple English translations of novels) and extracts pairs of dependency paths only from aligned sentences that share matching anchors. Anchors are allowed to be only nouns or pronouns, and they match if they are identical. Matching anchors become matched slots. Heuristic functions are used to score the anchor matches and the resulting template pairs, which are more likely to be paraphrases, rather than simply textual entailment pairs, since they are obtained from aligned sentences of a monolingual parallel corpus.

An alternative is to identify anchors in related sentences from comparable corpora, which are easier to obtain. Shinyama and Sekine (2003) [63] find pairs of sentences that share the same anchors within clusters of news articles reporting the same event. In their method, anchors are named entities (e.g. person names) identified using a named entity recognizer, or pronouns and noun phrases that refer to named entities. Dependency trees are then constructed from each pair of sentences, and pairs of dependency paths are extracted from the trees by treating anchors as slots.

### 2.5.2 Bootstrapping

Bootstrapping approaches can also be used in extraction, as in generation, but with the additional complication that there is no particular input template nor seed values of its slots to start from.

Brazilay and McKeown (2001) [15] used a bootstrapping method to extract paraphrases from a parallel monolingual corpus, involving two classifiers. One classifier examines the words the candidate paraphrases consist of, and a second one examines their contexts. The two classifiers use different feature sets and the output of each

classifier is used to improve the performance of the other one in an iterative manner. Words that occur in both sentences of an aligned pair are treated as positive lexical examples; all the other pairs of words from the two sentences become seed negative lexical examples. In each iteration, only k strongest positive and negative context rules are retrained. The strength of each context rule is its precision.

The context rules may also produce multi-word lexical examples, which are generalized by replacing their words by their POS tags. The paraphrasing rules are also filtered by their strength, which is the precision with which they predict paraphrasing contexts. The remaining paraphrasing rules are used to obtain more lexical examples, which are also filtered and so on until no new positive lexical examples can be obtained from the corpus, or a maximum number of iterations is exceeded.

### 2.5.3 Alignment

Barzilay and Lee (2003) [14] used comparable corpora. The sentence of each corpus were clustered separately and from each cluster a lattice was produced by aligning the cluster's sentences with Multiple Sequence Alignment [13]. Each sentence of a cluster corresponds to a path in the cluster's lattice. In each lattice, nodes that are shared by a high percentage (50% in Barzilay and Lee's experiments) of the cluster's sentences are considered backbone nodes. Parts of the lattice that connect otherwise consecutive backbone nodes are replaced by slots. If two slotted lattices from different corpora share many fillers, they are taken to be paraphrases. Hence, this method also uses the extended Distributional Hypothesis.

Pang et al.'s method (2003) [57] produces finite state automata, but it requires a parallel monolingual corpus. The parse trees of aligned sentences are constructed and then merged. Each merged tree is converted to a finite state automaton by traversing the tree in a depth-first manner and introducing a ramification when a node with a disjunction is encountered. All the language expressions that can be produced by the automaton are paraphrases.

Bannard and Callison-Burch (2005) [4] point out that bilingual parallel corpora are much easier to obtain and in much larger sizes, using statistical machine translation (SMT). More precisely, to paraphrase English phrases, they employ German as pivot language. They construct a phrase table from the parallel corpus, and from the table they estimate the probabilities P(e—f) and P(e—f), where e and f range over all of the English and pivot language phrases of the table. The best paraphrase $e_2^*$ of each English phrase $e_1$ in the table is computed as follows:

$$e_2^* = \arg\max P(e_2|e_1) = \arg\max \sum_{f \in T} P(f|e_1)P(e_2|f, e_1) \approx \arg\max \sum_{f \in T} P(f|e_1)P(e_2|f) \tag{2.5.1}$$

Multiple bilingual corpora, for different pivot languages, can be used, where C ranges over the corpora.

$$e_2^* = \arg\max \sum_{C} \sum_{f \in T(C)} P(f|e_1)P(e_2|f) \tag{2.5.2}$$

Zhao et al. (2008) [76] use a log-linear classifier to score candidate paraphrase pairs that share a common pivot phrase, instead of using equations. In effect, the classifier uses the above probabilities additionally to other features that assess the quality of the word alignment. In subsequent work, Zhao et al. (2009) [75] consider that two English phrases are paraphrases, when they are aligned to different pivot phrases.

### 2.5.4 Evaluation

As in generation, recall cannot be computed. Instead, one may again count the total yield of the method, possibly at different precision levels. Moreover, direct comparisons of extraction methods may be impossible. Furthermore, different scores are obtained, depending on whether the extracted pairs are considered in particular contexts or not, and whether they are required to be interchangeable grammatical sentences.

As in generation, in principle one could use a paraphrase recognizer to automatically score the extracted pairs. However, recognizers are not yet accurate enough;hence, human judges are usually employed. Again, one may also evaluate extraction methods indirectly, for example by measuring how much the extracted pairs help in information extraction [2].

## 2.6 Summary

Table 2.2 summarizes the main ideas discussed per task. The underlying ideas of generation and extraction methods are in effect the same, even if the methods perform different tasks; recognition work has relied on rather different ideas, quite similar to those of STS.

| main ideas discussed | R-P | G-P | E-P |
|---|---|---|---|
| Logic-based inferencing | X | | |
| Vector space semantic models | X | | |
| Surface string similarity measures | X | | |
| Syntactic similarity measures | X | | |
| Similarity measures on symbolic meaning representation | X | | |
| Machine learning algorithms | X | X | X |
| Decoding | X | X | |
| Word/sentence alignment | | X | X |
| Pivot language(s) | | X | X |
| Bootstrapping | | X | X |
| Distributional Hypothesis | | X | X |
| Sunchronous grammar rules | | X | X |

Table 2.2: Main ideas discussed and tasks they have mostly been used in. R: recognition, G: generation, E: extraction, P: paraphrasing

# Chapter 3

# Our Approach

In the previous chapter we saw that there are several ways in order to implement paraphrasing methods. There is a variety of similarity measures to be used and plenty ways to implement the alignment between two sentences. Our approach is focused on string similarity measures, which can capture the similarity or dissimilarity between two text strings. In parallel, we attempt to create alignment between two sentences by using Finite State Machines.

## 3.1 The idea

For the purposes of this thesis, we approach the problems of paraphrase recognition and detection by applying string similarity measures. The implementation of some of the measures precede the current work. Given the existing research by our lab members,[1] we have chosen to enrich it by adding word and character n-grams and creating a system, which is able to identify paraphrases effectively. Nevertheless, our approach can be extended and incorporate similarity measures derived from other sources (syntactic, knowledge-based, etc.).

The general idea for our alignment approach is fairly simple. We have used Finite State Machines (FSMs), in order to link a pair of sentences. In an attempt to increase the chances of a successful alignment, we have used english resources based on the google similarity[2]. By creating sets of semantically equivalent words we increase the possibility of the matching, especially if we refer to a specific-oriented domain. Based on that assumption, we implement a word-order sensitive similarity measure and perform paraphrase recognition. Furthermore, we use a language model, which can enrich our system with grammatically, that is to say the utterance of being grammatically well-formed. By adding the language model to our system, we target to generate paraphrases.

[1] Work of N. Malandrakis and V. Prokopi [45][46]
[2] Work of E. Iosif, PortDial project

## 3.2 String Similarity Metrics

In order to achieve our goal, we implemented a variety of string similarity metrics, which were applied to the text. Specifically, we used:

- ***Word n-grams and Character q-grams (n = 1,2,3 and q = 2,3)*** In the field of computational linguistics, an n-gram is a contiguous sequence of $n$ items from a given sequence of text.

  The concept of using n-grams is that the desired meaning is hidden inside each word or a sequence of words. In each case, we are sure that substrings of a sentence contain a significant amount of information and can be embedded in order to efficient approximate matching. Comparing character q-grams instead of word n-grams gives us the opportunity to distinguish similarity bewteen semantically equivalent words but not entirely identical. Character q-grams can capture similarity of words that derive from the same stem but differ on their suffix or prefix.

  The similarity is expressed as ([36]):

$$SIM(X,Y) = \frac{|X \cap Y|}{\alpha max(|X|,|Y|) + (1-\alpha)min(|X|,|Y|)} \tag{3.2.1}$$

  where $|X|$ and $|Y|$ are the number of n-grams of sentences X and Y respectively. The parameter $\alpha$ controls the effect of each sentence and it's no effect value is $\alpha = 0.5$.

  To build the SIM(X,Y) function we represent the sentence as sets of words or as sets of characters. Let's assume that we have the sentence *"A man with a hard hat is dancing."*. The word bigrams that will be produced are: {A man, man with, with a, a hard, hard hat, hat is, is dancing}, while the character bigrams for the word *"dancing"* will be {da, an, nc, ci, in, ng}.

- ***Longest Common Subsequence Similarity (LCSS)*** based on the Longest Common Subsequence (LCS) character based dynamic programming algorithm. LCSS represents the length of the longest string (or strings) that is a substring (or are substrings) of two or more strings.

  Let two sentences be defined as follows: $X = (x_1, x_2, ..., x_m)$ and $Y = (y_1, y_2, ..., y_n)$. The prefixes of X are $X_{1,2,...,m}$; the prefixes of Y are $Y_{1,2,...,n}$. Let $LCSS(X_i, Y_j)$ respresent the set of longest common subsequence of prefixes $X_i$ and $Y_j$. The function which defines the set of sequences is

$$LCSS(X_i,Y_j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCSS(X_{i-1}, Y_{j-1}) + 1 & \text{if } x_i = y_j \\ longest(LCSS(X_i, Y_{j-1}), LCSS(X_{i-1}, Y_j)) & \text{if } x_i \neq y_j \end{cases} \tag{3.2.2}$$

- ***Skip Bigrams*** coocurance measures the overlap of skip bigrams between two sentences or phrases. Skip-grams are a technique widely used, whereby n-grams are formed, but in addition to allowing adjacent sequence of words, to allow tokens to be "skipped". We define skip k-grams for a sentence $w_1...w_n$ to be the set

$$\{w_{i_1}, w_{i_2}, ..., w_{i_n} | \sum_{j=1}^{n} i_j - i_{j-1} < k\} \tag{3.2.3}$$

  Skip-grams reported for a certain skip distance (k) allow a total of k or less skips to construct the n-gram. As such, skip bigrams results include 1 skip and 0 skips.

  For example, let's assume the sentence *"A man with a hard hat is dancing."* and compare the bigrams and the skip-bigrams that are produced.
  Bigrams = {A man, man with, with a, a hard, hard hat, hat is, is dancing}.
  Skip-bigrams = {A man, A with, A a, man with, man a, man hard, with a, with hard, with hat, a hard, a hat, a is, hard hat, hard is, hard dancing, hat is, hat dancing, is dancing}.

- ***Containment metrics on n-grams (n = 1,2,3)*** It is defined as the percentage of a sentence that is contained in another sentence. It is a number between 0 and 1, where 1 means the hypothesis sentence is fully contained in the reference sentence. We express containment as the amount of n-grams of a sentence contained in another. The containment metric is not symmetric and is calculated as:

$$c(X, Y) = \frac{|S(X) \cap S(Y)|}{S(X)} \tag{3.2.4}$$

  where S(X) and S(Y) are all the n-grams of sentences X and Y respectively. [46]

- ***Number of n-gram matches and degree of similarity between non-matching words*** In order to compute the similarity between two sentences, a two-pass procedure was followed. First lexical hit rates were identified and then, the similarity was calculated among non-matched n-grams. [45] The semantic similarity scores from pairs of words are summed together in an attempt to obtain a score similar to the BLUE metric (Papineni et al., 2002). However, the alignment between the two input sentences is a problem. In order to cope with it, the comparison of the non-matched n-grams is employed from the first sentence to the second and vise versa.

## 3.3 Algorithm for Paraphrase Detection and Semantic Textual Similarity

Paraphrase detection, identifies a pair of sentences as paraphrases, returning a score, either 1 if the sentences are paraphrases or 0 if not. While, Semantic Textual Similarity (STS) computes a score, which ranges from 0 to 1, where 0 implies no similarity

between the two sentences and 1 for identical sentences.

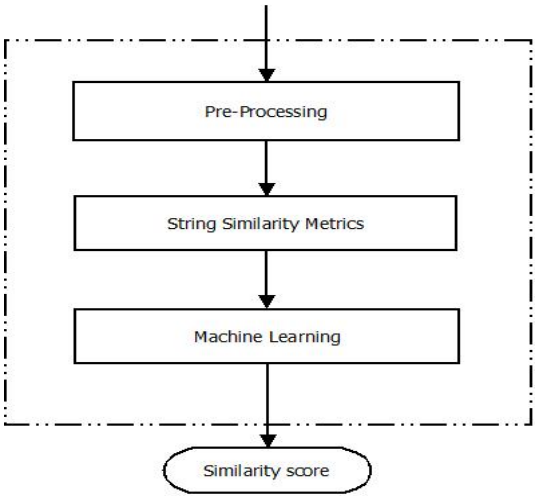



Figure 3.1: System architecture for Paraphrase detection and STS

According to the schematic representation of the algorithm, which is presented in figure (3.1), the system is comprised by 3 components.

The first component represents a pre-processing stage, in which we use the CoreNLP suite of tools (Finkel et al., 2005; Toutanova et al., 2003), a process that includes named entity recognition, normalization, part of speech tagging and lemmatization. Additionally, we achieve the stemming implementation by using the Porter Stemmer.

At the second stage, we applied to the input text the string similarity metrics. The computed similarity scores show the degree of semantic equivalence between a pair of sentences. The higher the score, the more identical the sentences are.

At the final component, we use the calculated similarity scores, to estimate the semantic similarity between the pairs of sentences contained in the corpus. In order to employ machine learning methods, we use the Weka Toolkit, which offers a variety of machine learning algorithms [3].

## 3.4 Algorithm for Paraphrase Recognition and Generation

At this section we present the algorithm constructed for paraphrase recognition and generation. Our work is basically based on an alignment implemented by Finite State

[3] `www.cs.waikato.ac.nz/ml/weka`

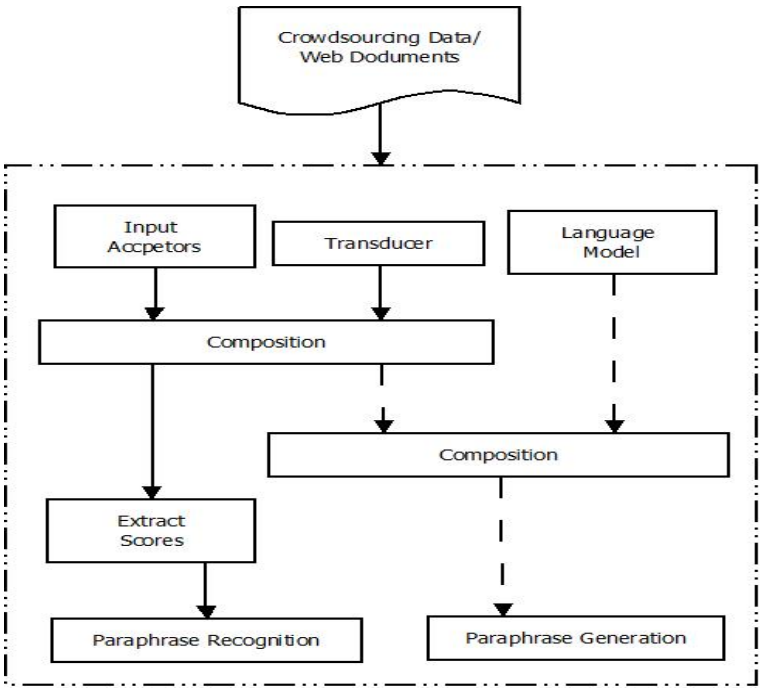



Figure 3.2: System architecture for Paraphrasing Prompts

Machines (FSMs). The schematic representation of the algorithm is presented in figure 3.2.

### 3.4.1 Baseline

As our baseline system we have used the word unigram feature, as it is defined in the 3.2.1 equation. However, the resulting similarity is based only on the shared words of the input pair, without preserving the word order inside the sentence.

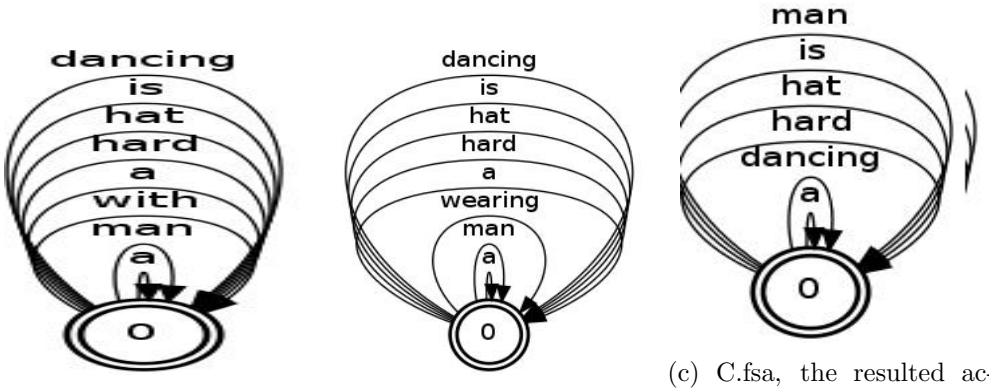
The recognition result will derive from the two input acceptors (A.fsa and B.fsa) in combination with the determinization of their intersection (C.fsa). So our system will correspond to:

$$C.fsa = A.fsa \cap B.fsa \tag{3.4.1}$$

The intersection function (defined with the symbol $\cap$) returns the common words of the two input acceptors with cost combined from the costs of the two acceptors. Moreover, the determinization of the resulting acceptor rejects the duplicate transitions.

For instance, let's assume as input pair the sentences $s_1$ and $s_2$, which correspond to *"A man with a hard hat is dancing."* and *"A man wearing a hard hat is dancing."* respectivelly.

The produced input and resulted FSAs would be:




(a) A.fsa, the acceptor for the input sentence $s_1$

(b) B.fsa, the acceptor for the input sentence $s_2$

(c) C.fsa, the resulted acceptor containing the shared words of the input FSAs

Figure 3.3: Example of FSAs for the baseline system

### 3.4.2 Word-order sensitive

Considering the implementation of the word unigrams metric as our baseline, we intended to add the concept of word-order to the computed similarity.

In our attempt to identify the semantic similarity between a pair of sentences we created a Finite State Acceptor (FSA) for each sentence, which accepts the sentence, its substrings and their concatenation. The recognition is employed by composing the FSAs with a Finite State Transducer (FST), which contains transitions based on the google similarities. An FST is a two-tape finite state machine, an input and an output, and has the ability to translate the contents of its input tape to the output.

In order to create an alignment between the two input acceptors, we created a transducer with identity mapping weight equal to zero and transitions for semantically equivalent words (synonyms) with weight based on the google similarity. More specifically, for every noun and verb contained in the sentences, which are included in a vocabulary of 135465 words, we extract the N synonyms with the highest google similarity score. After filtering the synonyms and rejecting those that have not the same part-of-speech (POS) tag as the original word, we conclude to a set of semantically equivalent words. We define the weight of the synonym transitions for the word $i$ as:

$$\{w_{i1}, w_{i2}, ..., w_{iN} | \sum_{j=1}^{N} 1 - G_{ij}\} \tag{3.4.2}$$

where $w_{ij}$ is the $j-th$ transition of the $i-th$ word, N the number of synomyms and G is a vector containing the N highest google similarities sorted in an decreasing order.

Finally, in order to avoid assymetry, the synonym transitions for the first sentence have as input the original word and as output the synonym, while for the second sentence, the order is inversed.

If we set the number of synonyms equal to zero (N=0), the system corresponds as intersecting the two input FSAs because the transducer will contain only word to word transitions. As a result, the produced FSA will be the concatenation of the common substrings of the input acceptors. The role of the transducer is to replace the epsilon transitions with synonyms. If an alignment exists, the epsilon transition will be replaced and the cost will change. If more than one alignments exist, then the one with the lowest cost will be chosen. Moreover, the POS filtering will maintain the semantic notion of the initial prompt.

So our recognition system will correspond to the best path of a composition of the input acceptor (A.fsa and B.fsa respectivelly) with a transducer (C.fst).

$$A.fsa \circ C.fst \circ B.fsa \tag{3.4.3}$$

The composition function (defined with the symbol $\circ$) returns a relational composition of the input FSMs in the given order. If an input machine is an acceptor, then it is treated as a transducer with identical input and output tape. Additionally, the best path function returns the lowest cost paths from the start state to the final one.

Except of using the best path, we employed paraphrase recognition by normalizing the cost extracted from the composition. The normalization employed is based on the sentences' length and its definition is:

$$Similarity(A.fsa, B.fsa) = 1 - \frac{C}{\alpha|A.fsa| + (1-\alpha)|B.fsa|} \tag{3.4.4}$$

where C is the resulting cost extracted from the composition of the input acceptors with the transducer, $|A.fsa|$ and $|B.fsa|$ are the length (in words) of the first and second sentence respectively and $\alpha$ is a parameter controlling the degree of the effect of each sentence in the similarity measure. It is set to its no effect value, equal to 0.5.

For the generation task, we used a different sequence of FSM functions. Initially, we have linked the resulting, from the composition, FSTs (D.fst) by using the union function (defined with the symbol $\bigcup$). Then, we compose the produced FST with an FSA created from a language model.

A statistical language model assigns probability to a sequence of $m$ words. The language model that we have used ia a task-dependent, smoothed back-off bigram in the .arpa format, with costs being $-0.3log_{10}(P(w))$. We created the .arpa file by using the CMU toolkit [4]. We convert it to a form compatible with fsm-toolkit and recalculate the costs, hence we manage to create an acceptor (LM.fsa).

Assigning weights was the main problem. The values we use in our experiments are the ones that produced the best results, however they are not considered as optimal.

So our generation system responds to the following definition:

[4] `http://www.speech.cs.cmu.edu/SLM_info.html`

$$(\bigcup D.fst) \circ LM.fsa \tag{3.4.5}$$

Let's assume the input pair $s_1$ and $s_2$, which refer to the sentences *"What date would you like to travel"* and *"What time do you want to fly"* and the acceptors 3.4 and 3.5 respectively. Each FSA has, additionally to the word, epsilon transitions with weights equal to 1, which corresponds to the acceptance of the sentence, its substrings and any concatenation of theirs. The weight was set to 1, in order to disfavour the deletions.

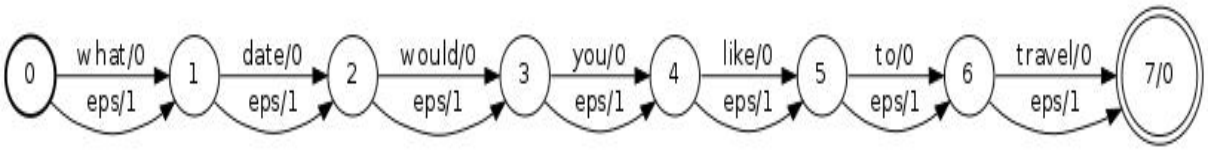



Figure 3.4: Acceptor, A.fsa, for the input sentence $s_1$ containing word and epsilon transitions.

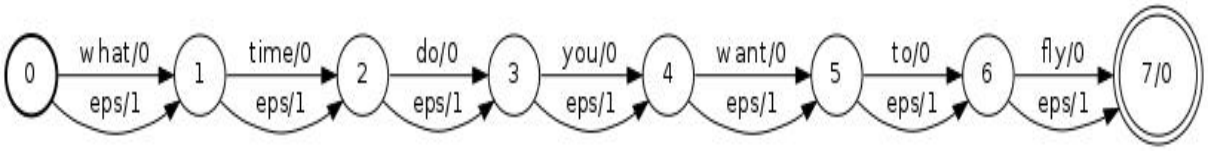



Figure 3.5: Acceptor, B.fsa, for the input sentence $s_2$ containing word and epsilon transitions.

The produced FST for the above example would be (3.6):

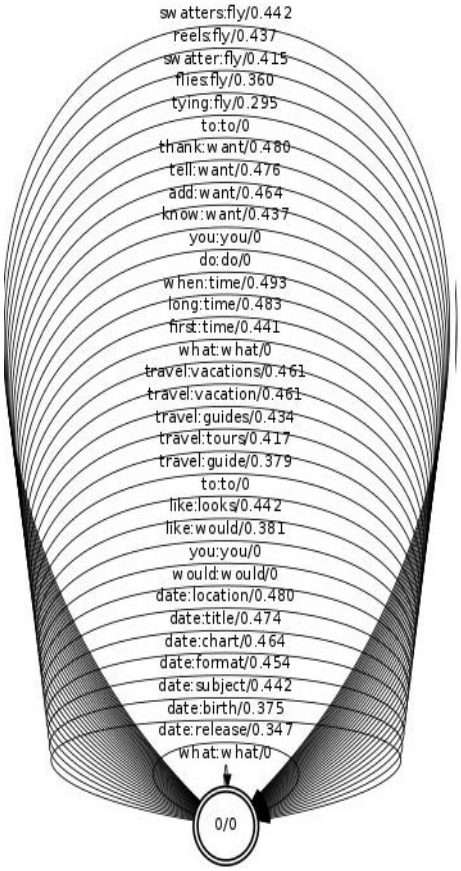



Figure 3.6: Example of a transducer, C.fst, for the input sentences $s_1$ and $s_2$, for a small number of synonyms (N).

For the recognition task, we have extracted the best path from the resulting FST (D.fst). Then, in order to extract a score for each sentence of the corpus, we have used the following sequence of fsm functions: project -2 and minimize (D.fsa) 3.7.

The project function returns the input or output of the transducer, depending on a parameter which controls the outcome. The parameter has value either 1 or 2 for the input and output tape respectively. The minimization of the resulting acceptor leads to a minimal deterministic acceptor.

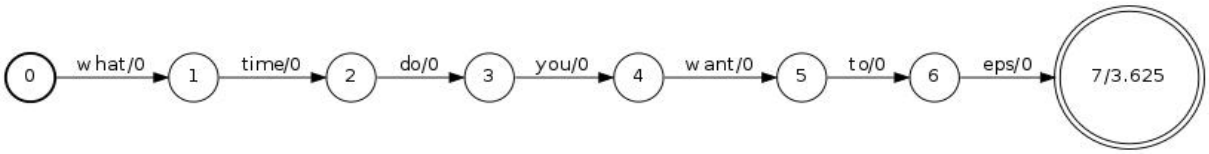



Figure 3.7: Resulting acceptor, D.fsa, after projecting and minimizing the composition $A.fsa \circ C.fst \circ B.fsa$ .

The language model is a back-off bigram model. For example the sentence $s_3$ "Do you need a hotel for tonight" it would be 3.8:

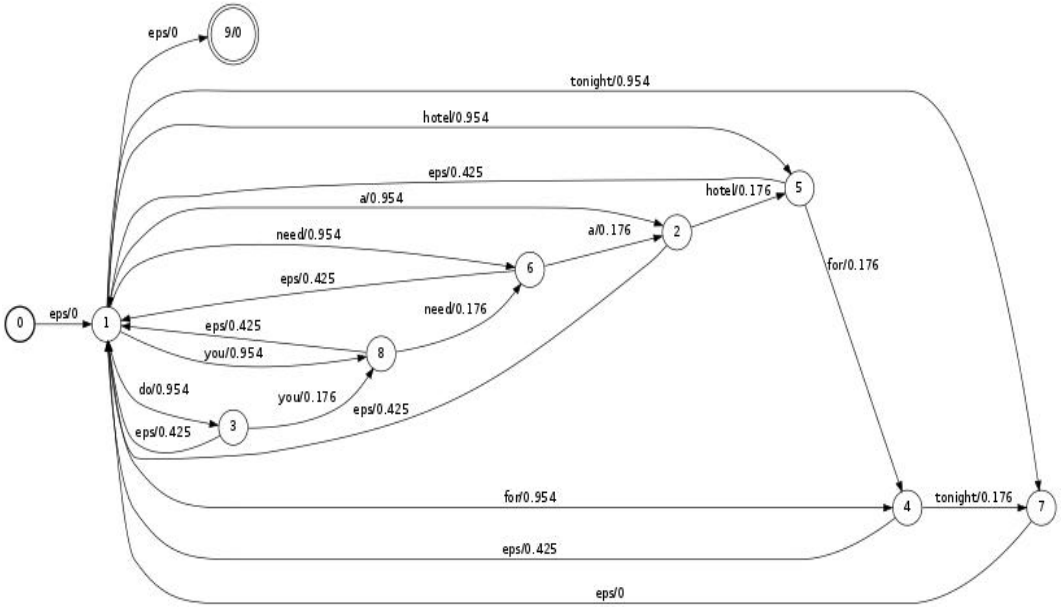



Figure 3.8: Language model, LM.fsa, produced from the sentence $s_3$

# Chapter 4

# Experimental Procedure & Evaluation

In this chapter we present the results of our work and describe our experiments for paraphrase detection, sementic textual similarity and paraphrasing methods, recognition and generation. Firstly, we evaluate the system based on string similarity features, which is used for paraphrase detection and semantic textual similarity (STS). Then, we present the experiments of our FSM-based algorithm. The best approach to show that our system works well would be to compare its performance with other systems. Unfortunately, in our case the comparison can not be performed due to the origin of the dataset. However, we evaluated the algorithm on the test datasets of SemEval 2012 and we present the results.

## 4.1 Evaluation on Paraphrase Detection

Paraphrase detection identifies a pair of sentences as paraphrases or not, returning a discrete value which characterizes each class.

For our experiments we have used the MSR paraphrase corpus, a benchmark dataset. It contains 5.801 pairs of sentences, obtained from clusters of online newsarticles referring to the same events. The sentence pairs were annotated by human judges as paraphrases or not. More information about the MSR paraphrase corpus are given in 2.3.8.

The training process includes two different classifiers, a Naive Bayes and an SVM classifier, although we ended using only the former. Their definition is described in 2.2.3.

For the evaluation step, we have selected four commonly used evaluation metrics, which are defined below: precision (equation 4.1.1), recall (equation 4.1.2), accuracy (equation 4.1.3) and F-measure (equation 4.1.4). Additionally, we have calculated the Pearson correlation coefficient. The definition of these measures is:

$$PR = \frac{TP}{TP + FP} \quad (4.1.1) \qquad RC = \frac{TP}{TP + FN} \quad (4.1.2)$$

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \quad (4.1.3) \qquad FM = \frac{2 * PR * RC}{PR + RC} \quad (4.1.4)$$

where :

- TP (true positives) are the numbers of pairs that have been correctly classified as positives (paraphrases).
- FP (false positives) are the numbers of pairs that have been incorrectly classified as positives.
- TN (true negatives) are the number of pairs that have been correctly classified as negatives (not paraphrases).
- FN (false negatives) are the number of pairs that have been incorrectly classified as negatives.

Additionally, we have choosen two techniques of estimating the performance. We employ either 10 fold-cross-validation or we use the 70% of the dataset for training and the 30% for testing.

***10 fold-cross-validation*** One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds.

| classiffier | NaiveBayes | | | | |
|---|---|---|---|---|---|
| evaluation metrics | accuracy | precision | recall | F-measure | correlation |
| Word Character Ngrams | **0.700** | 0.804 | **0.732** | **0.766** | **0.351** |
| LCS LCSstring skip2 containment | 0.647 | 0.811 | 0.620 | 0.703 | 0.303 |
| Ngram hits & Lex. non-matching words | 0.668 | 0.783 | 0.699 | 0.739 | 0.288 |
| All measures | 0.676 | **0.817** | 0.669 | 0.735 | 0.342 |

Table 4.1: Paraphrase Detection results, using a Naive Bayes classifier, without stemming implementation

| classiffier | NaiveBayes | | | | |
|---|---|---|---|---|---|
| evaluation metrics | accuracy | precision | recall | F-measure | correlation |
| Word Character Ngrams | **0.698** | 0.802 | **0.732** | **0.765** | **0.348** |
| LCS LCSstring skip2 containment | 0.647 | 0.811 | 0.620 | 0.703 | 0.303 |
| Ngram hits & Lex. non-matching words | 0.668 | 0.783 | 0.699 | 0.739 | 0.288 |
| All measures | 0.674 | **0.815** | 0.665 | 0.733 | 0.333 |

Table 4.2: Paraphrase Detection results, using a Naive Bayes classifier, with stemming implementation

The above tables contain scores for the three categories of metrics as described in the feature description 3.2.

The results clearly indicate that the combination of word and character n-grams perform better in four of our evaluation metrics. As we can see from the detailed tables, listed on the Appendix Results for Paraphrase Detection and STS, the main contributor for these results is the character n-gram metric, which achieves the highest performance, due to their ability to capture semantic similarity between non-identical words.

## 4.2 Evaluation on Semantic Textual Similarity

Semantic textual similarity estimates the degree of equivalence between a pair of sentences, returning a continuous score ranging from 0 to 1.

At the Semantic Textual Similarity task, we have used the test datasets of the SemEval workshop 2012 and 2013. The SemEval workshop 2012-2013 shared task of Semantic Textual Similarity (STS) copes with the semantic equivalence of sentences [1].

### 4.2.1 Datasets

Totally, the algorithm was tested at 9 datasets (4.2.1), all annotated by the AMT at a range from 0 to 5, where 0 implies no similarity, while 5 identical sentences.

***MSRpar*** Microsoft Research (MSR) has acquired two manually annotaded datasets. The first, called MSR Paraphrase (MSRpar) is widely used for semantic equivalence evaluation. After processing the dataset, the organizers sampled 1500 pairs of sentences, which were splitted 50% for training and 50% for testing.

***MSRvid*** The second dataset derived from the MSR is the MSR Video Paraphrase Corpus. The dataset was created by showing brief video segments to annotators from

[1]The participants were asked to estimate the semantic equivalence of the source datasets at a scale from 0 to 5. They could send a maximum of three system runs. After downloading the test datasets, they had a maximum of 120 hours to upload the results. After the submission deadline expired, the organizers published the gold standard in the task website, in order to ensure a transparent evaluation process. The gold standard evaluation, was employed by Amazon Mechanical Turk (AMT) in order to crowd source the annotation task.

Amazon Mechanical Turk (AMT), and ask them to provide an one sentence decription of the main action or event in the video. In total, 1500 pairs of sentences were sampled which were splitted 50% for training and 50% for testing.

***SMTeuroparl*** Given the strong connection between STS systems and Machine Translation evaluation metrics, the organizers selected pairs of sentences from the translation shared task of the 2007 and 2008 ACL Workshops on Statistical Machine Translation (WMT). This resulted in 729 unique training pairs. The test data is comprised of all Europarl human evaluated French-English pairs from WMT 2008.

***SMTnews*** One of the out-of-domain testing datasets of 2012 workshop. It comprised of all the human ranked French-English system submissions from the WMT 2007 news conversation test set, resulting in 351 unique system reference pairs.

***OnWN***$_{2012}$ The second surprise test dataset (consisting of 750 pairs of sentences), which is radically different, as it comprised 750 pairs of glosses from OntoNotes 4.0 and WordNet 3.1 senses. The similarity between the sense pairs was generated using simple word overlap.

***HDL*** The organizers used naturally occurring news headlines gathered by the Europe Media Monitor (EMM) engine from several different newssources. EMM clusters together related news. The goal was to generate a balanced data set across the different similarity ranges. Accordingly, 750 pairs of sentences were sampled.

***SMT*** The SMT dataset comprises pairs of sentences used in machine translation evaluation. The dataset was built based on two different 2 different sets depending on the evaluation metric. In total, it contains 750 pairs of sentences.

***OnWN***$_{2013}$ The OnWN dataset contains gloss pairs from OntoNotes-WordNet (OnWN). The pairs are sampled based on the string similarity ranging from 0.4 to 0.9. String similarity is used to measure the similarity between a pair of glosses. The OnWN set comprises 561 gloss pairs from OntoNotes 4.0 and WordNet 3.0.

***FnWN*** The FnWN dataset contains gloss pairs from FrameNet-WordNet (FnWN). The pairs are sampled based on the string similarity ranging from 0.4 to 0.9, as in the OnWN dataset. The FnWN set has 189 manually mapped pairs of senses from FrameNet 1.5 to WordNet 3.1. They are randomly selected from 426 mapped pairs.

### 4.2.2 Evaluation Results

Two evaluation metrics have been used, i.e. the Pearson and Spearman correlation coefficients, for every computed measure compared with the ground truth scores provided by the organizers of the SemEval workshop.

The two evaluation metrics we have chosen are widely used. Their definition is described in 2.2.4. As a second evaluation metric the weighted mean of Pearson and Spearman correlations is used. The Pearson and Spearman coefficient for each dataset is weighted according to the number of sentence pairs in that dataset.

$$WeightedMean = \frac{\sum_{i=1}^{5} r_i * n_i}{\sum_{i=1}^{5} n_i} \tag{4.2.1}$$

where $r_i$ is the Pearson/Spearman scores for each dataset and $n_i$ the number of pairs in each dataset.

In order to compute the correlation between the similarites calculated by the features and the gold standards provided by the organizers of the SemEval, the Weka toolkit was used. More specifically, we used 3 classifiers, a Linear Regression classifier, a Bagging classifier and an M5P classifier. Their definition is described in 2.2.3.

- ***Linear Regression classiffier***

| Linear Regression | | | | | | |
|---|---|---|---|---|---|---|
| | MSRpar | MSRvid | SMTeuroparl | OnWN | SMTnews | Mean |
| Word Character Ngrams | **0.597** | 0.753 | 0.552 | 0.715 | 0.501 | **0.644** |
| LCS LCSstring skip2 containment | 0.373 | 0.516 | 0.448 | 0.672 | 0.164 | 0.464 |
| Ngram hits & Lex. non-matching words | 0.449 | 0.673 | 0.443 | 0.238 | 0.299 | 0.432 |
| All measures | 0.494 | **0.814** | **0.572** | **0.726** | **0.510** | 0.641 |

Table 4.3: Evaluation of string similarity measures on the test datasets of SemEval 2012 using a Linear Regressor

| Linear Regression | | | | | |
|---|---|---|---|---|---|
| | FNWN | headlines | OnWN | SMT | Mean |
| Word Character Ngrams | **0.440** | 0.717 | **0.550** | 0.334 | 0.524 |
| LCS LCSstring skip2 containment | 0.129 | 0.696 | 0.510 | 0.330 | 0.480 |
| Ngram hits & Lex. non-matching words | 0.089 | 0.692 | 0.171 | 0.307 | 0.383 |
| All measures | 0.356 | **0.745** | 0.519 | **0.371** | **0.531** |

Table 4.4: Evaluation of string similarity measures on the test datasets of SemEval 2013 using a Linear Regressor

- ***Bagging classiffier***

| Bagging | | | | | | |
|---|---|---|---|---|---|---|
| | MSRpar | MSRvid | SMTeuroparl | OnWN | SMTnews | Mean |
| Word Character Ngrams | 0.572 | 0.752 | 0.551 | 0.702 | 0.532 | 0.639 |
| LCS LCSstring skip2 containment | 0.531 | 0.649 | 0.578 | 0.656 | 0.573 | 0.602 |
| Ngram hits & Lex. non-matching words | 0.479 | 0.678 | 0.480 | 0.295 | 0.311 | 0.461 |
| All measures | **0.588** | **0.812** | **0.610** | **0.724** | **0.598** | **0.679** |

Table 4.5: Evaluation of string similarity measures on the test datasets of SemEval 2012 using a Bagging classiffier

| Bagging | | | | | |
|---|---|---|---|---|---|
| | FNWN | headlines | OnWN | SMT | Mean |
| Word Character Ngrams | 0.330 | 0.708 | 0.640 | 0.318 | 0.529 |
| LCS LCSstring skip2 containment | 0.165 | 0.674 | 0.630 | **0.341** | 0.509 |
| Ngram hits & Lex. non-matching words | 0.050 | 0.685 | 0.245 | 0.313 | 0.398 |
| All measures | **0.343** | **0.724** | **0.698** | 0.318 | **0.550** |

Table 4.6: Evaluation of string similarity measures on the test datasets of SemEval 2013 using a Bagging classiffier

- ***M5P classiffier***

| M5P | | | | | | |
|---|---|---|---|---|---|---|
| | MSRpar | MSRvid | SMTeuroparl | OnWN | SMTnews | Mean |
| Word Character Ngrams | 0.597 | 0.752 | **0.552** | **0.712** | 0.488 | 0.642 |
| LCS LCSstring skip2 containment | 0.555 | 0.587 | 0.453 | 0.672 | 0.453 | 0.563 |
| Ngram hits & Lex. non-matching words | 0.451 | 0.689 | 0.443 | 0.238 | 0.301 | 0.437 |
| All measures | **0.617** | **0.811** | 0.531 | 0.676 | **0.561** | **0.658** |

Table 4.7: Evaluation of string similarity measures on the test datasets of SemEval 2012 using an M5P classiffier

| M5P | | | | | |
|---|---|---|---|---|---|
| | FNWN | headlines | OnWN | SMT | Mean |
| Word Character Ngrams | **0.440** | 0.717 | 0.568 | **0.334** | **0.529** |
| LCS LCSstring skip2 containment | 0.090 | 0.696 | 0.588 | **0.334** | 0.498 |
| Ngram hits & Lex. non-matching words | 0.089 | 0.692 | 0.180 | 0.317 | 0.389 |
| All measures | 0.332 | **0.746** | **0.699** | 0.183 | 0.512 |

Table 4.8: Evaluation of string similarity measures on the test datasets of SemEval 2013 using an M5P classiffier

- Comparing the classifiers.

| | datasets | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| classifiers | MSRpar | MSRvid | SMTeuroparl | ONWN | SMTnews | FNWN | headlines | OnWN | SMT |
| LinearRegression | 0.494 | **0.814** | 0.572 | **0.726** | 0.510 | **0.356** | 0.745 | 0.519 | **0.371** |
| Bagging | 0.588 | 0.812 | **0.610** | 0.724 | **0.598** | 0.343 | 0.724 | 0.698 | 0.318 |
| M5P | **0.617** | 0.811 | 0.531 | 0.676 | 0.5621 | 0.332 | **0.746** | **0.699** | 0.183 |

Table 4.9: The results show the correlation of "All measures" as described abobe among all datasets (both SemEval 2012 and 2013) and all the classifiers

Tables 4.3, 4.4, 4.5, 4.6, 4.7 and 4.8 contain scores for every category of metrics as described in 3.2 and their combination, while we change the classiffier and the applied dataset. The combination of all measures achieves in most cases the highest performance. Character and word n-grams are quite competitive against the combination of all measures, considering that they achieve quite high correlation scores. However, the applied dataset contributes considerably to the performance.

Table 4.9 shows the correlation scores of the "All measures" which contains all the metrics. The idea is to compare the system's performance and the classiffier's contribution in it. As we can see, the performance varies among the datasets. For datasets with small sentences, such as MSRvid and headlines, the Linear Regression classifier seems to achieve equally high scores with the other classifiers, while in other datasets its performance reduces significantly.

## 4.3 Experiments on Paraphrasing Prompts

At this section we present the results for paraphrase recognition and generation, with an FSM-based algorithm. The proposed model has been tested in two kinds of travel-domain corpora, in order to paraphrase prompts. The first corpus contains data collected from the CrowdFlower crowdsoursing servise, while the second is harvested from the web.

We have categorized the results according to the input dataset. The first section refers to data collected from crowdsoursing while the second to web documents. Results are listed in detail at the Results for Paraphrasing Prompts

### 4.3.1 Data Collection

Initially, we wanted to create datasets containing travel domain information. In order to achieve our goal, we used web documents and data collected from the CrowdFlower crowdsoursing servise.

#### Crowdsoursing Data

To create this corpus, we collected data from the CrowdFlower crowdsoursing servise [2]. Users were asked to fill a questionnaire with 4 tasks. At the first task, they were asked to create a question, given an answer and the opposite. At the second task, users were asked to paraphrase parts of dialogues, while the third was to complete a dialogue. Finally, at the last task, users were asked to fill in blanks from dialogues.

Gathering the data from the first and the third task, we manage to create a dataset with prompts. Even though the specific corpus was significantly more relevant than the web documents, we employed the same pre-processing steps:

1. 5 to 15 words per sentence. After analyzing datasets containing prompts from the PortDial project, we concluded that the majority of prompts contain 5 to 15 words, with exception prompts which refer to more than one category. For example, asking about departure date and destination in one sentence.

2. at least 3 mutual words with a list of keywords. This list contains words such as date, travel, hotel etc. This criterion was established in order to reject sentences that do not refer to the travel domain.

3. make the sentences of the corpus unique.

4. finally, the corpus contained 433 prompts, covering the categories of: PLACE, DATE, DAY, TIME.

[2] Work of E. Palogiannidi [56]

**Web Documents**

At first, we used the "human-human" corpus from the PortDial project (`www.portdial.eu`) datasets in order to extract travel domain prompts. The following dialogue was contained in the specific dataset. We refer to the System's answers as prompts.

*SYSTEM: AND WHAT DAY IN MAY DID YOU WANT TO TRAVEL*
*USER: OKAY I NEED TO BE THERE FOR A MEETING THAT'S FROM THE TWELFTH TO THE FIFTEENTH*
*SYSTEM: AND FLYING INTO WHAT CITY*
*USER: SEATTLE*
*SYSTEM: AND WHAT TIME WOULD YOU LIKE TO LEAVE PITTSBURGH*
*USER: I DON'T THINK THERE'S MANY OPTIONS FOR NON-STOP*
*SYSTEM: RIGHT THERE'S THREE NON-STOPS A DAY*

After extracting the prompts, we categorized them depending on their subject, resulting 7 categories: PLACE, AIRPORT, DATE, TIME, DAY, HOTEL, CAR. The next step was to create queries. Totally, we used 55 queries, which we used in order to harvest web documents from the web.

| CATEGORIES | EXAMPLE |
|---|---|
| DATE | what date would you like to travel |
| TIME | what time would you like to leave |
| DAY | what date did you need to depart |
| PLACE | flying into what city |
| AIRPORT | which airport would you like to fly to |
| CAR | did you need a car |
| HOTEL | what hotel would you like |

Table 4.10: Categories of prompts and examples

However, the data collected had to be processed and cleaned. For that purpose, we considered the following characteristics as mandatory:

1. 5 to 15 words per sentence. After analyzing datasets containing prompts from the PortDial project, we concluded that the majority of prompts contain 5 to 15 words, with exception prompts which refer to more than one category. For example, asking about departure date and destination at one sentence.

2. at least 3 mutual words with a list of keywords. This list contains words such as date, travel, hotel etc. This criterion was established in order to reject sentences that do not refer to the travel domain.

3. make the sentences of each dataset unique.

4. finally, 7 datasets have been created, one for each category.

| DATASET | SIZE (in sentences) |
|---|---|
| DATE | 77 |
| TIME | 314 |
| DAY | 113 |
| PLACE | 857 |
| AIRPORT | 136 |
| CAR | 381 |
| HOTEL | 352 |

Table 4.11: Size of each web dataset

### 4.3.2 Results on Crowdsourcing Data

For the input prompt "WHAT DATE WOULD YOU LIKE TO TRAVEL", we show the best 10 candidate paraphrases according to the cost, normalized or not, that was calculated for each sentence of the corpus.

| Input prompt | WHAT DATE WOULD YOU LIKE TO TRAVEL | |
|---|---|---|
| | without normalization | with normalization |
| Best Candidate Paraphrases | what time would you like to travel<br>what date do you want to travel<br>what date would you like<br>what date would you like to depart<br>what date would you like to leave<br>what day would you like to travel<br>what date do you like to leave<br>what date did you wish to travel<br>what time of day would you like to travel<br>what time would you like to depart | what time would you like to travel<br>what date do you want to travel<br>what day would you like to travel<br>what date would you like to leave<br>what date would you like to depart<br>what time of day would you like to travel<br>what date would you like to book a flight<br>what date would you like<br>what date do you like to leave<br>what date did you wish to travel |

Table 4.12: Candidate paraphrases for the input prompt: WHAT DATE WOULD YOU LIKE TO TRAVEL.

The recognition results on the crowdsourcing data are quite good, regardless the existance of the normalization factor on the produced scores. As we can see from the table 4.12, the 10 best candidate paraphrases maintain the semantic concept of the input prompt.

By adding the language model, we manage to generate paraphrases. The best 10 produced paraphrases for the same input prompt are:

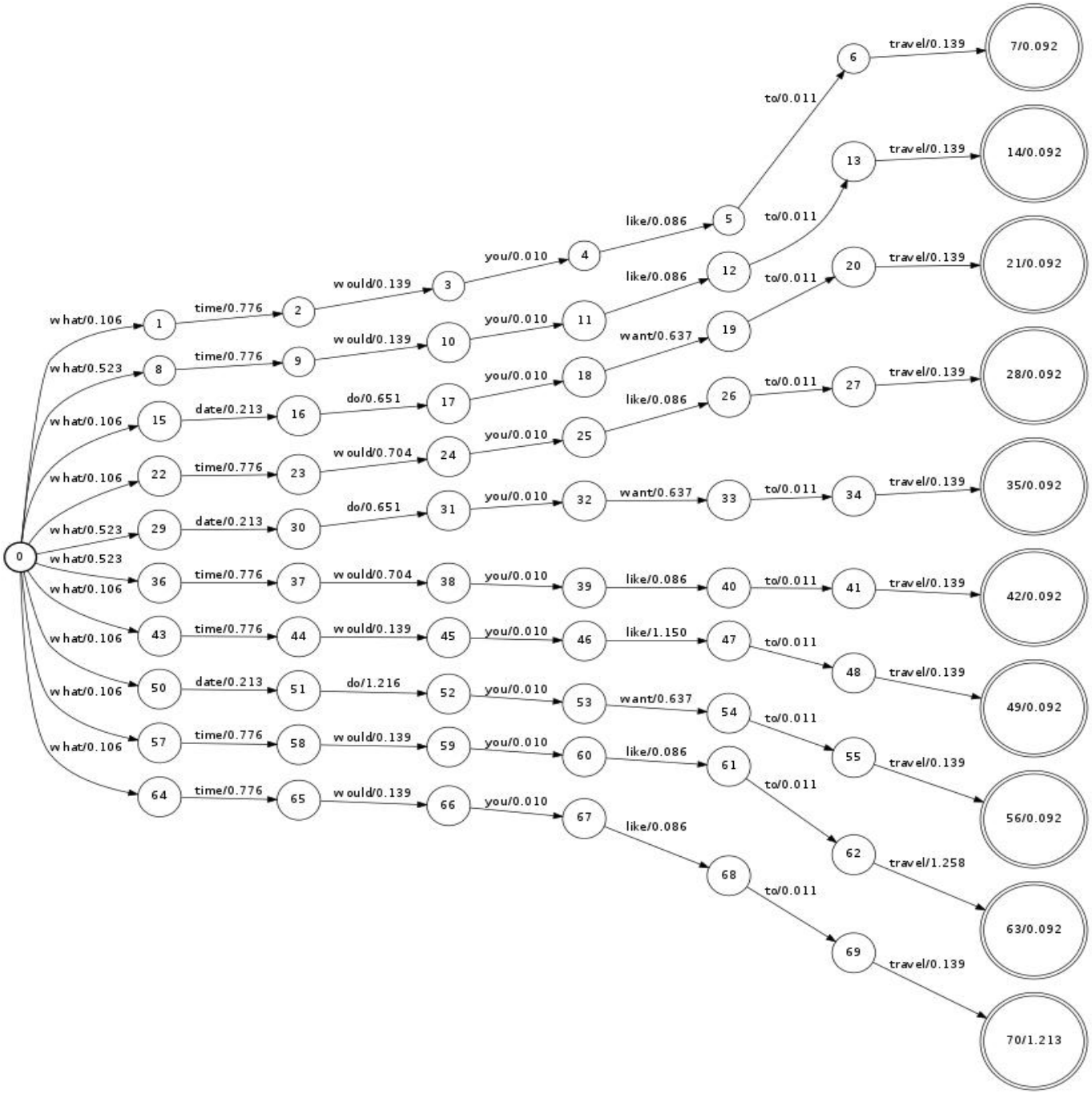



Figure 4.1: Candidate Paraphrases for input prompt “WHAT DATE WOULD YOU LIKE TO TRAVEL”

The generation results are good but not as impressive as in the recognition task. The language model contributes to that, since the weights used are the ones that give us premium but not optimal results.

### 4.3.3 Results on Web Documents

For the input prompt "WILL YOU NEED A HOTEL", we show the best 10 candidate paraphrases according to the cost, normalized or not, that was calculated for each sentence of the corpus.

| Input prompt | WILL YOU NEED A HOTEL | |
|---|---|---|
| | without normalization | with normalization |
| Best Candidate Paraphrases | do you need a hotel | do you need a hotel |
| | but do you need a hotel | why do you need a hotel |
| | do you need a hotel job | where do you need a hotel |
| | do you need a hotel pickup | when do you need a hotel |
| | do you need a hotel rental | do you need a hotel tonight |
| | do you need a hotel reservation | do you need a hotel roommate |
| | do you need a hotel room | do you need a hotel room |
| | do you need a hotel roommate | do you need a hotel reservation |
| | do you need a hotel tonight | do you need a hotel rental |
| | when do you need a hotel | do you need a hotel pickup |

Table 4.13: Candidate paraphrases for the input prompt: WILL YOU NEED A HOTEL.

The above results were extracted specifically from the dataset referring to HOTEL. The results for the rest of the datasets (AIRPORT, CAR, DATE, DAY, PLACE, TIME) will be found in the Results for Paraphrasing Prompts.

The recognition results in the case of web documents are quite good (table 4.13). However, the quality of the information contained on the documents is to be questioned.

By adding the language model, we manage to generate paraphrases. The best 10 produced paraphrases for the same input prompt are:

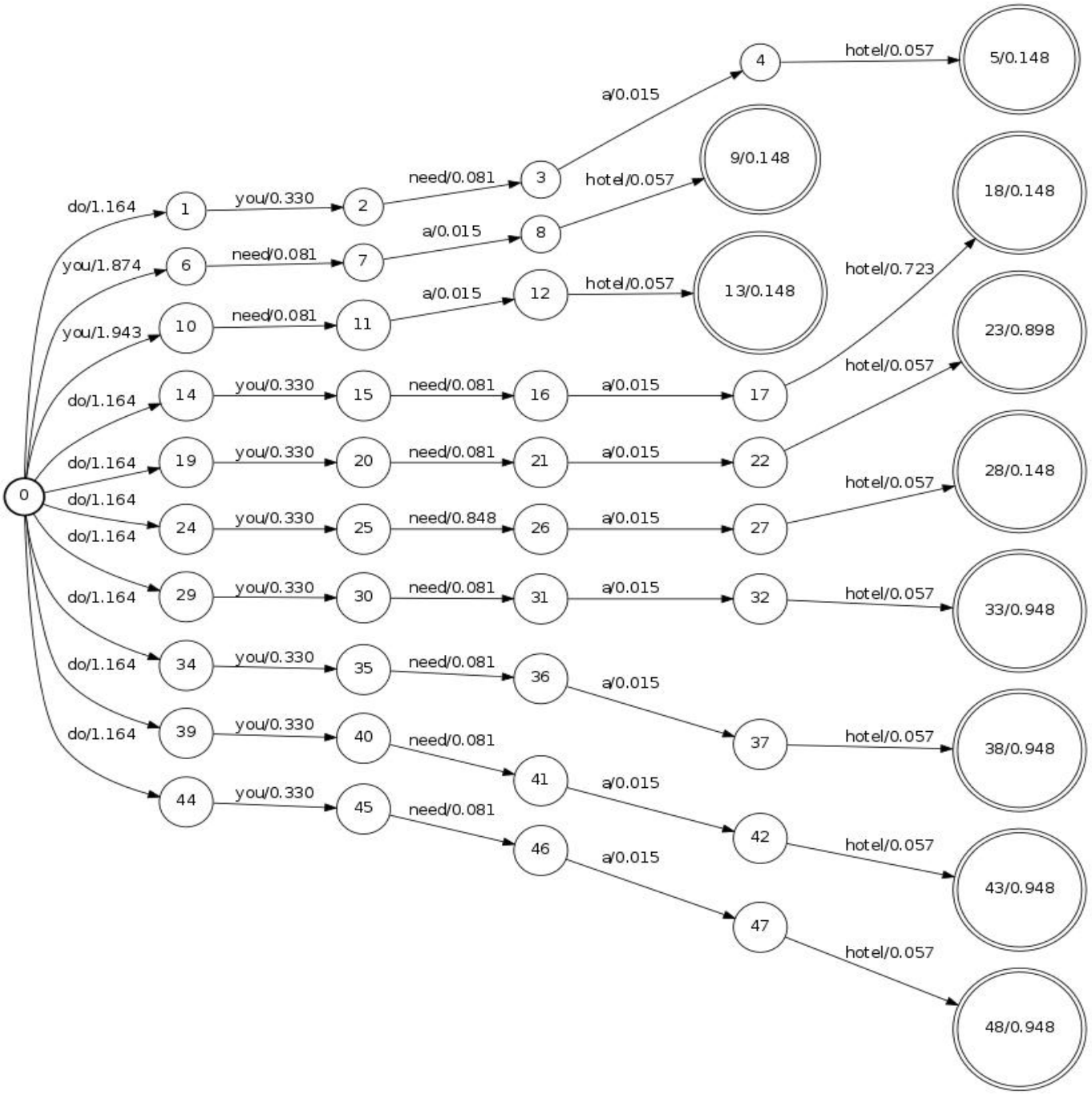



Figure 4.2: Candidate Paraphrases for input prompt “WILL YOU NEED A HOTEL”

The web documents contain, except of prompts, sentences that have either small or no semantic similarity with our target domain, resulting a poor language model, which effects the generation results.

### 4.3.4 Objective Evaluation

In order to evaluate our approach, we managed to estimate the semantic equivalence among texts. For this purpose, we tested our algorithm on the test datasets of the SemEval workshop of 2012 (MSRpar, MSRvid, SMTeuroparl, SMTnews, OnWN). Their description is on 4.2.1.

#### Baseline system

The simplest version of the algorithm, which is the FSM implementation of the word unigram feature. The produced FSA accepts the shared words of the two input sentences or any combination of them, without preserving the word order.

The evaluation of the above system shows results equal to the one computed for the word unigram similarity feature, as the score produced are based on the number of the unique words either of the input sentences or of their intersection.

| | Datasets | | | | |
|---|---|---|---|---|---|
| Evaluation Measure | MSRpar | MSRvid | SMTeuroparl | OnWN | SMTnews |
| Pearson correlation | 0.539 | 0.443 | 0.498 | 0.661 | 0.412 |
| Spearman correlation | 0.512 | 0.471 | 0.559 | 0.666 | 0.408 |

Table 4.14: Evaluation scores of the baseline system

#### Word-order sensitive FSMs

At this point, we attempt to experiment on the two parameters of the algorithm, the number of semantically equivalent words (N) and the cost of the epsilon transitions (e).

For the evaluation we have computed the Perason and Spearman correlation coefficient for every combination of the two parameters, cost of epsilon transition in the input FSA and number of synonyms (N).

| | | Datasets | | | | |
|---|---|---|---|---|---|---|
| Number of synonyms (N) | epsilon cost (e) | MSRpar | MSRvid | SMTeuroparl | OnWN | SMTnews |
| N = 0 | e = 0.2 | 0.397 | 0.374 | 0.485 | 0.635 | 0.399 |
| | e = 0.4 | 0.397 | 0.374 | 0.485 | 0.635 | 0.399 |
| | e = 0.6 | 0.397 | 0.374 | 0.485 | 0.635 | 0.399 |
| | e = 0.8 | 0.397 | 0.374 | 0.485 | 0.635 | 0.399 |
| | e = 1 | 0.397 | 0.374 | 0.485 | 0.636 | 0.398 |
| N = 20 | e = 0.2 | 0.398 | 0.376 | 0.485 | 0.636 | 0.398 |
| | e = 0.4 | 0.406 | 0.371 | 0.491 | 0.640 | 0.404 |
| | e = 0.6 | 0.411 | 0.367 | 0.492 | 0.640 | 0.405 |
| | e = 0.8 | 0.413 | 0.363 | 0.492 | 0.641 | 0.404 |
| | e = 1 | 0.414 | 0.360 | 0.492 | 0.641 | 0.402 |
| N = 40 | e = 0.2 | 0.398 | 0.376 | 0.485 | 0.636 | 0.398 |
| | e = 0.4 | 0.408 | 0.382 | 0.491 | 0.641 | 0.404 |
| | e = 0.6 | 0.414 | 0.385 | 0.491 | 0.643 | 0.404 |
| | e = 0.8 | 0.417 | 0.385 | 0.491 | 0.643 | 0.404 |
| | e = 1 | 0.418 | 0.385 | 0.490 | 0.643 | 0.402 |
| N = 60 | e = 0.2 | 0.398 | 0.376 | 0.485 | 0.636 | 0.398 |
| | e = 0.4 | 0.407 | 0.390 | 0.490 | 0.643 | 0.408 |
| | e = 0.6 | 0.413 | 0.400 | 0.490 | 0.646 | 0.413 |
| | e = 0.8 | 0.416 | 0.403 | 0.488 | 0.647 | 0.414 |
| | e = 1 | 0.416 | 0.404 | 0.487 | 0.647 | 0.413 |
| N = 80 | e = 0.2 | 0.398 | 0.376 | 0.485 | 0.636 | 0.398 |
| | e = 0.4 | 0.406 | 0.395 | 0.489 | 0.644 | 0.408 |
| | e = 0.6 | 0.410 | 0.407 | 0.488 | 0.647 | 0.412 |
| | e = 0.8 | 0.412 | 0.412 | 0.486 | 0.648 | 0.413 |
| | e = 1 | 0.413 | 0.414 | 0.485 | 0.648 | 0.412 |

Table 4.15: Pearson correlation coefficient on the datasets of the SemEval workshop of 2012. Experimenting on the two parameters, number of synonyms (N) and cost of the epsilon transition (e).

By changing the values of the parameters, number of synonyms (N) and the cost of the epsilon transition (e), we conclude that increasing "e", achieves better performance. However, increasing the parameter N, for N=40, N=60 and N=80, does not show such an improvement, as shown in Table 4.15.

In an attempt to improve our system's performance, we have combined the scores derived from 4 versions of it, using 3 different classifiers. For the machine learning, the Weka toolkit was used and especially a Linear Regression, a Bagging and an M5P classifier. To implement this idea, we used scores from the baseline system (3.4.1), the word-order sensitive FSM (3.4.2) for N = 0, 20, where N is the number of synonyms according to the google similarity. Additionally, we used an implementation of the FSM system, which preserves the word order, however it does not filter the synonyms according to their POS (part-of-speech) tag. Finally, for all the above versions of the algorithm, except the baseline system, the cost for the epsilon transition is set to 1.

| | Datasets | | | | |
|---|---|---|---|---|---|
| classifiers | MSRpar | MSRvid | SMTeuroparl | OnWN | SMTnews |
| Linear Regressor | **0.548** | 0.461 | 0.492 | **0.657** | **0.387** |
| Bagging | **0.548** | 0.457 | 0.502 | 0.653 | 0.379 |
| M5P | 0.501 | **0.534** | **0.579** | 0.639 | 0.374 |

Table 4.16: Correlation on the datasets of SemEval 2012 after combining systems with different parameters, using a Linear Regressor, a Bagging and an M5P classifier.

Comparing the tables 4.14, 4.15 and 4.16, we can see that the combination of the FSM-based approaches presents higher correlation scores than our main system (table 4.15, referring to word-order sensitive FSMs 3.4.2). However, the correlation is still lower than the baseline system (table 4.14, referring to the FSM-implemented word unigram measure 3.4.1).

Finally, we selected the SMTnews dataset to experiment on by increasing the number of semantically equivalent words (N,synonyms), based on the google similarity. The parameter N was set to 250 and 500, i.e. quite larger than the previous experiments.

| | Number of synonyms (N) | |
|---|---|---|
| epsilon cost (e) | N = 250 | N = 500 |
| e = 0.2 | 0.398 | 0.398 |
| e = 0.4 | 0.407 | 0.407 |
| e = 0.6 | 0.411 | 0.409 |
| e = 0.8 | 0.413 | 0.410 |
| e = 1 | 0.412 | 0.408 |

Table 4.17: Correlation on the SMTnews dataset of SemEval after increasing the number of synonyms to N = 250 and N = 500

The correlation is lower than that of the baseline system, although the difference is quite small. That is expected since we have added the word-order constraint in our calculated similarity.

# Chapter 5

# Conclusions and Future Work

## 5.1 Conclusions

Overall, the completion of this work led to several valuable conclusions. First of all, we managed to discover the meaning of natural language only by using the surface information of a text, without the support of grammatical rules.

We created a baseline system with string similarity metrics in order to detect paraphrases and estimate semantic textual similarity. For both tasks, we used widely known datasets to conclude that our proposed metrics perform well comparing to larger systems.

The best results are mainly produced by the combination of all metrics, while the character n-gram feature shows quite impressive performance. The obtained results showed that character n-grams can capture the semantic similarity between non-identical words. Additionally, we tried to introduce some additional constraints in order to improve our results, so we added length normalization to our features.

Our second goal was to create a system that is able to recognize and generate paraphrases via a composition of Finite State Machines (FSMs). For the experimental procedure, we focused on the travel domain and more specifically on prompts. The corpora we used were either harvested from the web or collected from the CrowdFlower crowdsourcing servise.

The main factor of this approach is the usage of English resources based on the google similarity, in order to build an alignment between two sentences. The recognizer achieves quite good results, while the generator shows fair but not spectacular performance.

We built a bigram language model using the CMU toolkit. The contribution of the language model to the generator was the factor that defined our results. In the case of the web documents the outcome has weak, mostly because of the unrelevant information included in the documents. However, for the data collected from crowdsourcing, the results seem promising. Improvements in this section should benefit the system's performance.

## 5.2 Future Work

### 5.2.1 Similarity Features

This thesis approached the paraphrase detection and semantic textual similarity problem at the level of string similarity features. An obvious improvement would be to develop a larger system comprised of similarity features which would evolve the existing one. Including not only string similarity features but context or knowledge-based metrics (for instance WordNet) may improve our performance.

Further improvement could be gained by including part-of-speech information, in order to measure syntactic constraints, in the similarity feaures.

### 5.2.2 FSM-based alignment

Moreover, improvement could come from the task of prompts paraphrasing. Experimenting more with the parameters, the number of semantically equivalent words (N) and the cost of the epsilon transition, could give us a clearer view regarding their contribution to both recognizing and generating prompts. Increasing the parameter N will, theoretically, show use better results, although that is not certain.

Additionally, electing to substitute google similarity for a similarity based on context may improve the existing alignment procedure. Discarding the semantically equivalent words used for the alignment with different criteria, such as the word's sense, can create alignments more relevant to the travel domain.

### 5.2.3 Data Collection

We have selected to test our algorithm on travel-domain corpora, either harvested from the web or collected from crowsoursing. The quality of the included information is ambiguous, especially in the case of web documents.

Presently, the language model is task-dependent, as indicated by the fairly good results in the generating task. Creating a language model based on a more relevant corpus or experimenting on the transition weights may alleviate the problem.

# Appendix A

# The FSM Toolkit

## Overview

The AT&T FSM library [1] is a set of general-purpose software tools available for Unix, for building, combining, optimizing, and searching weighted finite-state acceptors and transducers. Finite-state transducers are automata for which each transition has an output label in addition to the more familiar input label. Weighted acceptors or transducers are acceptors or transducers in which each transition has a weight as well as the input or input and output labels. The original goal of the AT&T FSM library was to provide algorithms and representations for phonetic, lexical, and language-modeling components of large vocabulary speech recognition systems.
This imposed the following requirements:

1. Generality: to support the representation and use of the various information sources in speech recognition

2. Modularity: to allow rapid experimentation with different representations

3. Efficiency: to support competitive large-vocabulary recognition using automata of more than 10 million states and transitions.

The mathematical foundation of the library is the theory of rational power series, which supplies the semantics for the objects and operations and creates opportunity for optimizations such as determinization and minimization. System Components:

1. AT&T FSM library: includes about 30 stand-alone commands to construct, combine, determinize, minimize, and search weighted finite-state machines (FSMs). These commands manipulate FSMs by reading from and writing to files or pipelines.

2. Dot and Dotty: programs used by the FSM library to visualize graph representations of FSMs (Graphviz).

[1] `www.research.att.com/~fsmtools/fsm/man3/fsm.1.html`

# Commands

1. **FSM COMPILATION AND DISPLAY**

***fsmcompile*** takes input representing an FSM from file file or standard input, and sends to standard output its binary encoding. The input should be the textual representation of an FSM. FSM states, input symbols and output symbols are represented in the input by non-negative numbers, unless the options -s symbols, -i symbols, -o symbols are used. These options allow state, input symbols and output symbols, respectively, to be given textual names, where symbols files give the translation from those names and numbers . The input should be an acceptor, unless the -t option is given, in which case it should be a transducer.

***fsmprint*** prints the input FSM on standard output using same textual format as fsmcompile accepts as input. States, input symbols and output symbols are printed in numeric form, unless the options -s symbols, -i symbols, -o symbols are used to provide textual names for states, input symbols and output symbols, respectively.

***fsmdraw*** sends to standard output a dot(1) graph representation of the input FSM (The command dot -Tps can be used to convert from dot format to PostScript.) States, input symbols and output symbols are displayed in numeric form, unless the options -s symbols, -i symbols, -o symbols are used 56 to provide textual names for states, input symbols and output symbols, respectively. The options -w x and -h x set the page width and height (in inches), -f fontname sets the font name (default is Times-Roman), -F n sets the font size (in points), -p use portrait mode (default is landscape), and -v displays vertically (i.e., top-to-bottom; default is left-to-right).

2. **FSM CONSTRUCTION AND COMBINATION**

***fsmunion*** returns the union of one or more input FSMs.

***fsmconcat*** returns the concatenation of one or more input FSMs, in the order specified by the command-line arguments.

***fsmclosure*** returns the Kleene closure of the input FSM. With the -p option, the empty string is not added, that is, Kleene + is used instead of Kleene *.

***fsmrmepsilon*** returns an equivalent FSM with no epsilon transitions. The input FSM must have no negative cost epsilon cycles.

***fsmintersect*** returns the intersection of two or more acceptors. Each input FSM accepts string s iff the output FSM accepts s with the costs combined by

the EXTEND operation.

***fsmcompose*** returns the relational composition of the input FSMs, in the order given in the command line. With two input FSMs, for example, if the first machine transduces string s1 to s2 and the second machine transduces s2 to s3, then the output machine will transduce s1 to s3 with the two costs combined by the EXTEND operation. If an input machine is an acceptor, it is treated as a transducer from the language it accepts to itself.

***fsmdifference*** returns the intersection of the acceptor fsm1 with the complement of the costless, deterministic, epsilon-free acceptor fsm2.

3. **FSM MINIMIZATION AND EQUIVALENCE**

***fsmconnect*** returns an FSM from which any states and arcs in the input that do not lie on a path from the start state to a final state have been removed. With the -t option, it returns exit status 1 if the output has no states, which is useful for testing the input for emptiness.

***fsmdeterminize*** determinizes the input FSM, which must be determinizable. Epsilon arcs are treated the same as other symbols.

***fsmminimize*** returns the minimal deterministic FSM equivalent to the input FSM, which must be a deterministic acceptor. Epsilon arcs are treated the same as other symbols.

***fsmarccollect*** COLLECTS costs on identically-labelled arcs between the same source and destination states.

***fsmcompact*** uses a heuristic procedure to return an FSM equivalent to fsm. but possibly smaller. It works for arbitrary FSMs.

***fsmequiv*** exits with zero status if fsm1 and fsm2 are equivalent. The inputs must be deterministic, epsilon-free acceptors.

4. **FSM SEARCH**

***fsmbestpath*** returns the lowest-cost path from the start state of the input FSM to a final state. The path is encoded as a (single path) FSM. With the -n nbest option, the nbest lowest-cost paths are returned. The output 58 is encoded as an FSM that is the union of the individual paths in increasing cost order. With

the -c cthresh and -N nthresh options, the input FSM is pruned as in fsmprune, limiting the nbest search. With the -u option, all paths returned will be distinct strings.

***fsmprune*** returns those states and arcs that lie on paths whose total path cost in fsm is within cthresh of the lowest cost path and at most the nthresh best such states. input epsilons. In each case, if there is a cycle with respect to the sorting criterion, fsmtopsort returns the input FSM unsorted.

***fsminfo*** sends to standard output the following information about the input FSM its FSM class and whether it is an acceptor or transducer. With the -n option, various numeric information is printed, including the number of states, number of transitions, final states, epsilon transitions, strongly-connected components, accessible states, and co-accessible states. With the -p option, FSMProps is called on the input, which will return pre-computed information about the FSM, such as whether it is cyclic, costless, non-negative, or deterministic. If precomputed information about a property is not supported by the FSM class, a ? is printed for it. With the -t option, values for all FSM properties are printed (by explicit tests run on the FSM if needed). See fsmprops.h for the set of defined FSM properties. With the -c option, the FSM class properties are printed, which include the FSM operations supported by that class. See fsmprops.h for the set of defined FSM class properties. With the -q w option, quantiles in intervals of width w are printed for various data including state in-degree, state out-degree, input label, output label, and arc cost. With the -b q1 option, the quantiles begin at q1 (default: 0.0), and with the -e q2 option, the quantiles end at p2 (default: 1.0). The -v option is equivalent to -tcn -q4.

# Appendix B

# Results for Paraphrase Detection and STS

The Character n-gram features, used in both Paraphrase Detection and Semantic Textual Similarity, are computed via the following equations:

$$SIM(A,B) = \frac{|A \cap B|}{\alpha max(|A|,|B|) + (1-\alpha)min(|A|,|B|)} \tag{B.1}$$

$$SIM(A,B) = \log\left(\frac{|A \cap B| + b}{|A| + |B|}\right) \tag{B.2a}$$

$$SIM(A,B) = \log\left(\frac{|A \cap B| + b}{|A| * |B|}\right) \tag{B.2b}$$

$$SIM(A,B) = \frac{|A \cap B| + b}{|A| * |B|} \tag{B.2c}$$

$$SIM(A,B) = e^{(-2*G_0)} \tag{B.2d}$$

$$G_0 = \frac{max(\log(|A|),\log(|B|)) - \log(|A \cap B|)}{\log(\Delta) - min(\log(|A|),\log(|B|))} \tag{B.2e}$$

where |A |and |B |is the number of bigrams of the the first and the second sentence, respectively.

At the first equation the parameter a controls the effect of each sentence and its no-effect value is $\alpha = 0.5$. The parameter b is set to its no-effect value (b=0) and describes the degree to which all of the words have commonalities among each other.

At the last equation $\Delta = 15000$, under the assumption that the average number of character n-grams in a word is 6 and the average sentence length is 25 words.

For the features LCS, LCSstring and Skip bigrams, two types of normalization have been employed:

$$NormSimilarity = \frac{Similarity}{(|A| + |B|)} \tag{B.3}$$

$$NormSimilarity = \frac{Similarity}{(|A| * |B|)} \tag{B.4}$$

The containment n-grams feature has been computed for unigrams, bigrams and trigrams (n = 1,2,3). The normalization for these measures is:

$$NormSimilarity = \frac{Similarity}{|A|} \tag{B.5}$$

$$NormSimilarity = \frac{Similarity}{|B|} \tag{B.6}$$

At the following section, the results for each mertic's performance and their combinations in both tasks are listed. We have categorized the results depending on the machine learning employed and the existence of stemming or not. For the machine learning, we have used two classifiers, a NaiveBayes and an SVM.

## Paraphrase Detection

The tables below show the results of each metric on the MSR paraphrase dataset for the task of detection.

The combination of Word and Character n-grams with the string similarity measures achieve the highest scores almost in every case.

Two baselines are included, $BASE_1$ classifies all pairs as paraphrases; $BASE_2$ classifies two sentences as paraphrases when their surface word edit distance is below a threshold.

| NaiveBayes | | | | | |
|---|---|---|---|---|---|
| measures | accuracy | precision | recall | F-measure | correlation |
| character bigrams B.1 | 0.722 | 0.754 | 0.870 | 0.808 | 0.325 |
| bigrams B.2a | 0.722 | 0.745 | 0.892 | 0.812 | 0.314 |
| bigrams B.2b | 0.677 | 0.677 | 0.994 | 0.805 | 0.092 |
| bigrams B.2c | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| bigrams B.2d | **0.726** | 0.752 | 0.883 | 0.812 | 0.327 |
| character trigrams B.1 | 0.719 | 0.755 | 0.862 | 0.805 | 0.321 |
| trigrams B.2a | 0.722 | 0.745 | 0.892 | 0.812 | 0.313 |
| trigrams B.2b | 0.673 | 0.675 | 0.992 | 0.803 | 0.046 |
| trigrams B.2c | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| trigrams B.2d | 0.718 | 0.748 | 0.877 | 0.807 | 0.307 |
| word 1grams B.1 | 0.726 | 0.761 | 0.862 | 0.809 | 0.338 |
| word 2grams B.1 | 0.686 | 0.715 | 0.885 | 0.791 | 0.204 |
| word 3grams B.1 | 0.680 | 0.693 | 0.941 | 0.798 | 0.142 |
| LCS B.3 | 0.701 | 0.729 | 0.882 | 0.799 | 0.258 |
| LCS B.4 | 0.679 | 0.686 | 0.963 | 0.801 | 0.123 |
| LCSstring B.3 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| LCSstring B.4 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| SKIP2 B.3 | 0.344 | 0.765 | 0.035 | 0.067 | 0.016 |
| SKIP2 B.4 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| containment 1gram | 0.720 | 0.779 | 0.815 | 0.797 | 0.349 |
| containment 2gram | 0.677 | 0.750 | 0.781 | 0.765 | 0.254 |
| containment 3gram | 0.657 | 0.735 | 0.766 | 0.750 | 0.204 |
| character 2grams | 0.716 | 0.785 | 0.796 | 0.790 | **0.353** |
| character 3grams | 0.707 | 0.784 | 0.779 | 0.782 | 0.340 |
| bigrams | 0.711 | 0.793 | 0.772 | 0.782 | 0.352 |
| trigrams | 0.704 | 0.789 | 0.763 | 0.776 | 0.341 |
| word Ngrams | 0.679 | 0.784 | 0.721 | 0.751 | 0.302 |
| character Ngrams | 0.706 | 0.795 | 0.758 | 0.776 | 0.350 |
| LCS | 0.698 | 0.748 | 0.831 | 0.787 | 0.278 |
| LCSstring | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| SKIP2 | 0.344 | 0.765 | 0.035 | 0.067 | 0.016 |
| containment | 0.668 | 0.788 | 0.693 | 0.737 | 0.297 |
| Word Character Ngrams | 0.700 | 0.804 | 0.732 | 0.766 | 0.351 |
| LCS LCSstring skip2 containment | 0.647 | 0.809 | 0.622 | 0.703 | 0.296 |
| allMeasures | 0.678 | **0.815** | 0.674 | 0.738 | 0.339 |
| $BASE_1$ | 0.665 | 0.665 | 1.000 | 0.799 | |
| $BASE_2$ | 0.69 | 0.724 | 0.863 | 0.788 | |
| Appendix D: | | | | | |

Table B.1: Evaluation on paraphrase detection, using a Naive Bayes classifier. No stemming has been employed while 10 fold-cross-validation has been used

| SVM | | | | | |
|---|---|---|---|---|---|
| measures | accuracy | precision | recall | F-measure | correlation |
| character bigrams B.1 | 0.722 | 0.744 | 0.896 | 0.813 | 0.313 |
| bigrams B.2a | 0.724 | 0.740 | 0.909 | 0.816 | 0.312 |
| bigrams B.2b | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| bigrams B.2c | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| bigrams B.2d | 0.721 | 0.735 | 0.916 | 0.815 | 0.299 |
| character trigrams B.1 | 0.721 | 0.744 | 0.893 | 0.812 | 0.307 |
| trigrams B.2a | 0.712 | 0.726 | 0.918 | 0.811 | 0.276 |
| trigrams B.2b | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| trigrams B.2c | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| trigrams B.2d | 0.713 | 0.731 | 0.908 | 0.810 | 0.282 |
| word 1grams B.1 | 0.725 | 0.753 | 0.879 | 0.811 | 0.331 |
| word 2grams B.1 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| word 3grams B.1 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| LCS B.3 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| LCS B.4 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| LCSstring B.3 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| LCSstring B.4 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| SKIP2 B.3 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| SKIP2 B.4 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| containment 1gram | 0.727 | 0.753 | 0.885 | 0.813 | 0.333 |
| containment 2gram | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| containment 3gram | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| bigrams | 0.726 | 0.737 | 0.921 | 0.819 | 0.312 |
| trigrams | 0.711 | 0.719 | 0.934 | 0.813 | 0.264 |
| character 2grams | 0.724 | 0.736 | 0.921 | 0.818 | 0.310 |
| character 3grams | 0.709 | 0.717 | 0.938 | 0.813 | 0.260 |
| word Ngrams | 0.734 | 0.759 | 0.885 | 0.817 | 0.350 |
| character Ngrams | 0.722 | 0.729 | 0.935 | 0.819 | 0.302 |
| LCS | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| LCSstring | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| SKIP2 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| containment | 0.738 | 0.758 | 0.897 | 0.821 | 0.357 |
| Word Character Ngrams | 0.743 | 0.761 | 0.901 | 0.825 | 0.369 |
| LCS LCSstring skip2 containment | 0.742 | 0.764 | 0.892 | 0.823 | 0.370 |
| allMeasures | **0.752** | **0.773** | 0.894 | 0.829 | **0.398** |
| $BASE_1$ | 0.665 | 0.665 | 1.000 | 0.799 | |
| $BASE_2$ | 0.69 | 0.724 | 0.863 | 0.788 | |

Table B.2: Evaluation on paraphrase detection, using a SVM classifier. No stemming has been employed while 10 fold-cross-validation has been used)

| NaiveBayes | | | | | |
|---|---|---|---|---|---|
| measures | accuracy | precision | recall | F-measure | correlation |
| character bigrams B.1 | 0.718 | 0.752 | 0.867 | 0.805 | 0.314 |
| bigrams B.2a | 0.719 | 0.743 | 0.888 | 0.809 | 0.301 |
| bigrams B.2b | 0.681 | 0.681 | 0.986 | 0.806 | 0.110 |
| bigrams B.2c | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| bigrams B.2d | 0.716 | 0.744 | 0.881 | 0.807 | 0.303 |
| character trigrams B.1 | 0.718 | 0.754 | 0.861 | 0.804 | 0.316 |
| trigrams B.2a | 0.718 | 0.741 | 0.893 | 0.810 | 0.299 |
| trigrams B.2b | 0.675 | 0.678 | 0.985 | 0.803 | 0.070 |
| trigrams B.2c | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| trigrams B.2d | 0.716 | 0.746 | 0.876 | 0.806 | 0.303 |
| word 1grams B.1 | **0.724** | 0.758 | 0.866 | 0.809 | 0.333 |
| word 2grams B.1 | 0.687 | 0.719 | 0.879 | 0.791 | 0.213 |
| word 3grams B.1 | 0.681 | 0.694 | 0.939 | 0.798 | 0.145 |
| LCS B.3 | 0.708 | 0.742 | 0.866 | 0.800 | 0.284 |
| LCS B.4 | 0.681 | 0.687 | 0.964 | 0.803 | 0.126 |
| LCSstring B.3 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| LCSstring B.4 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| SKIP2 B.3 | 0.350 | 0.767 | 0.048 | 0.091 | 0.022 |
| SKIP2 B.4 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| containment 1gram | 0.720 | 0.777 | 0.819 | 0.797 | 0.346 |
| containment 2gram | 0.677 | 0.751 | 0.777 | 0.764 | 0.255 |
| containment 3gram | 0.656 | 0.738 | 0.758 | 0.748 | 0.208 |
| bigrams | 0.708 | 0.792 | 0.768 | 0.780 | 0.349 |
| trigrams | 0.700 | 0.790 | 0.754 | 0.772 | 0.335 |
| character 2grams | 0.714 | 0.786 | 0.791 | 0.788 | 0.348 |
| character 3grams | 0.705 | 0.785 | 0.773 | 0.779 | 0.338 |
| word Ngrams | 0.677 | 0.781 | 0.723 | 0.751 | 0.296 |
| character Ngrams | 0.707 | 0.797 | 0.756 | 0.776 | 0.353 |
| LCS | 0.703 | 0.753 | 0.829 | 0.790 | 0.292 |
| LCSstring | 0.673 | 0.681 | 0.968 | 0.799 | 0.085 |
| SKIP2 | 0.350 | 0.767 | 0.048 | 0.091 | 0.022 |
| containment | 0.671 | 0.789 | 0.697 | 0.740 | 0.302 |
| Word Character Ngrams | 0.698 | 0.802 | 0.732 | 0.765 | **0.348** |
| LCS LCSstring skip2 containment | 0.655 | 0.804 | 0.643 | 0.715 | 0.299 |
| allMeasures | 0.680 | **0.812** | 0.682 | 0.741 | 0.337 |
| $BASE_1$ | 0.665 | 0.665 | 1.000 | 0.799 | |
| $BASE_2$ | 0.69 | 0.724 | 0.863 | 0.788 | |

Table B.3: Evaluation on paraphrase detection, using a Naive Bayes classifier. Stemming has been employed while 10 fold-cross-validation has been used

| SVM | | | | | |
|---|---|---|---|---|---|
| measures | accuracy | precision | recall | F-measure | correlation |
| character bigrams B.1 | 0.719 | 0.741 | 0.894 | 0.810 | 0.302 |
| bigrams B.2a | 0.720 | 0.736 | 0.912 | 0.814 | 0.299 |
| bigrams B.2b | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| bigrams B.2c | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| bigrams B.2d | 0.720 | 0.736 | 0.911 | 0.814 | 0.298 |
| character trigrams B.1 | 0.718 | 0.740 | 0.894 | 0.810 | 0.299 |
| trigrams B.2a | 0.713 | 0.727 | 0.919 | 0.812 | 0.277 |
| trigrams B.2b | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| trigrams B.2c | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| trigrams B.2d | 0.715 | 0.731 | 0.911 | 0.811 | 0.277 |
| word 1grams B.1 | 0.727 | 0.754 | 0.881 | 0.813 | 0.330 |
| word 2grams B.1 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| word 3grams B.1 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| LCS B.3 | 0.682 | 0.683 | 0.985 | 0.807 | 0.100 |
| LCS B.4 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| LCSstring B.3 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| LCSstring B.4 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| SKIP2 B.3 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| SKIP2 B.4 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| containment 1gram | 0.727 | 0.750 | 0.889 | 0.814 | 0.329 |
| containment 2gram | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| containment 3gram | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| bigrams | 0.723 | 0.735 | 0.919 | 0.817 | 0.299 |
| trigrams | 0.708 | 0.716 | 0.939 | 0.812 | 0.252 |
| character 2grams | 0.724 | 0.736 | 0.920 | 0.818 | 0.308 |
| character 3grams | 0.708 | 0.716 | 0.939 | 0.812 | 0.251 |
| word Ngrams | 0.735 | 0.756 | 0.893 | 0.819 | 0.347 |
| character Ngrams | 0.720 | 0.727 | 0.934 | 0.818 | 0.296 |
| LCS | 0.695 | 0.706 | 0.937 | 0.805 | 0.201 |
| LCSstring | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| SKIP2 | 0.672 | 0.672 | 1.000 | 0.804 | NaN |
| containment | 0.735 | 0.755 | 0.897 | 0.820 | 0.351 |
| Word Character Ngrams | 0.743 | 0.760 | 0.902 | 0.825 | 0.364 |
| LCS LCSstring skip2 containment | 0.741 | 0.762 | 0.894 | 0.823 | 0.366 |
| allMeasures | **0.746** | **0.767** | 0.894 | 0.825 | **0.378** |
| $BASE_1$ | 0.665 | 0.665 | 1.000 | 0.799 | |
| $BASE_2$ | 0.69 | 0.724 | 0.863 | 0.788 | |

Table B.4: Evaluation on paraphrase detection, using a SVM classifier. Stemming has been employed while 10 fold-cross-validation has been used

| NaiveBayes | | | | | |
|---|---|---|---|---|---|
| measures | accuracy | precision | recall | F-measure | correlation |
| character bigrams B.1 | 0.721 | 0.754 | 0.872 | 0.808 | 0.345 |
| bigrams B.2a | 0.723 | 0.746 | 0.894 | 0.813 | 0.325 |
| bigrams B.2b | 0.680 | 0.680 | 0.993 | 0.807 | 0.094 |
| bigrams B.2c | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| bigrams B.2d | **0.724** | 0.751 | 0.885 | 0.813 | 0.343 |
| character trigrams B.1 | 0.718 | 0.755 | 0.863 | 0.805 | 0.335 |
| trigrams B.2a | 0.720 | 0.743 | 0.895 | 0.812 | 0.337 |
| trigrams B.2b | 0.675 | 0.677 | 0.995 | 0.806 | 0.084 |
| trigrams B.2c | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| trigrams B.2d | 0.714 | 0.745 | 0.876 | 0.805 | 0.344 |
| word 1grams B.1 | 0.722 | 0.760 | 0.859 | 0.807 | 0.369 |
| word 2grams B.1 | 0.684 | 0.713 | 0.889 | 0.791 | 0.231 |
| word 3grams B.1 | 0.679 | 0.693 | 0.943 | 0.799 | 0.185 |
| LCS B.3 | 0.705 | 0.739 | 0.871 | 0.800 | NaN |
| LCS B.4 | 0.684 | 0.691 | 0.963 | 0.805 | NaN |
| LCSstring B.3 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| LCSstring B.4 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| SKIP2 B.3 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| SKIP2 B.4 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| containment 1gram | 0.722 | 0.781 | 0.817 | 0.799 | 0.354 |
| containment 2gram | 0.677 | 0.752 | 0.778 | 0.765 | 0.267 |
| containment 3gram | 0.655 | 0.735 | 0.765 | 0.750 | 0.227 |
| bigrams | 0.709 | 0.793 | 0.769 | 0.781 | 0.358 |
| trigrams | 0.702 | 0.792 | 0.758 | 0.775 | 0.352 |
| character 2grams | 0.718 | 0.789 | 0.795 | 0.792 | 0.362 |
| character 3grams | 0.707 | 0.788 | 0.774 | 0.781 | 0.357 |
| word Ngrams | 0.678 | 0.787 | 0.718 | 0.751 | 0.296 |
| character Ngrams | 0.700 | 0.796 | 0.747 | 0.771 | 0.370 |
| LCS | 0.698 | 0.750 | 0.829 | 0.788 | NaN |
| LCSstring | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| SKIP2 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| containment | 0.670 | 0.794 | 0.689 | 0.738 | 0.257 |
| Word Character Ngrams | 0.693 | 0.803 | 0.723 | 0.761 | **0.373** |
| LCS LCSstring skip2 containment | 0.660 | 0.805 | 0.656 | 0.723 | 0.299 |
| allMeasures | 0.680 | **0.814** | 0.682 | 0.742 | 0.344 |
| $BASE_1$ | 0.665 | 0.665 | 1.000 | 0.799 | |
| $BASE_2$ | 0.69 | 0.724 | 0.863 | 0.788 | |

Table B.5: Evaluation on paraphrase detection, using a Naive Bayes classifier. No stemming has been employed while we use 70% of the dataset as a training sample and 30% as testing

| SVM | | | | | |
|---|---|---|---|---|---|
| measures | accuracy | precision | recall | F-measure | correlation |
| character bigrams B.1 | 0.723 | 0.738 | 0.914 | 0.817 | 0.329 |
| bigrams B.2a | 0.723 | 0.731 | 0.933 | 0.820 | 0.346 |
| bigrams B.2b | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| bigrams B.2c | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| bigrams B.2d | 0.716 | 0.728 | 0.923 | 0.814 | 0.334 |
| character trigrams B.1 | 0.717 | 0.734 | 0.910 | 0.813 | 0.303 |
| trigrams B.2a | 0.710 | 0.720 | 0.934 | 0.813 | 0.271 |
| trigrams B.2b | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| trigrams B.2c | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| trigrams B.2d | 0.711 | 0.726 | 0.918 | 0.811 | 0.304 |
| word 1grams B.1 | 0.723 | 0.748 | 0.891 | 0.813 | 0.355 |
| word 2grams B.1 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| word 3grams B.1 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| LCS B.3 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| LCS B.4 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| LCSstring B.3 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| LCSstring B.4 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| SKIP2 B.3 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| SKIP2 B.4 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| containment 1gram | 0.726 | 0.752 | 0.888 | 0.814 | 0.349 |
| containment 2gram | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| containment 3gram | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| bigrams | 0.723 | 0.732 | 0.930 | 0.819 | 0.346 |
| trigrams | 0.710 | 0.718 | 0.939 | 0.814 | 0.288 |
| character 2grams | 0.721 | 0.731 | 0.927 | 0.817 | 0.345 |
| character 3grams | 0.709 | 0.717 | 0.939 | 0.813 | 0.284 |
| word Ngrams | 0.735 | 0.757 | 0.894 | 0.820 | 0.365 |
| character Ngrams | 0.718 | 0.724 | 0.940 | 0.818 | 0.338 |
| LCS | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| LCSstring | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| SKIP2 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| containment | 0.736 | 0.757 | 0.896 | 0.821 | 0.374 |
| Word Character Ngrams | 0.741 | 0.759 | 0.903 | 0.825 | 0.383 |
| LCS LCSstring skip2 containment | 0.742 | 0.762 | 0.899 | 0.825 | 0.365 |
| allMeasures | **0.751** | **0.773** | 0.894 | 0.829 | **0.411** |
| $BASE_1$ | 0.665 | 0.665 | 1.000 | 0.799 | |
| $BASE_2$ | 0.69 | 0.724 | 0.863 | 0.788 | |

Table B.6: Evaluation on paraphrase detection, using a SVM classifier. No stemming has been employed while we use 70% of the dataset as a training sample and 30% as testing

| NaiveBayes | | | | | |
|---|---|---|---|---|---|
| measures | accuracy | precision | recall | F-measure | correlation |
| character bigrams B.1 | 0.718 | 0.752 | 0.869 | 0.806 | 0.321 |
| bigrams B.2a | 0.718 | 0.743 | 0.891 | 0.810 | 0.324 |
| bigrams B.2b | 0.681 | 0.683 | 0.985 | 0.807 | 0.153 |
| bigrams B.2c | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| bigrams B.2d | 0.717 | 0.745 | 0.883 | 0.808 | 0.312 |
| character trigrams B.1 | 0.719 | 0.755 | 0.864 | 0.806 | 0.323 |
| trigrams B.2a | 0.719 | 0.741 | 0.896 | 0.811 | 0.308 |
| trigrams B.2b | 0.676 | 0.678 | 0.989 | 0.805 | 0.116 |
| trigrams B.2c | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| trigrams B.2d | 0.719 | 0.749 | 0.878 | 0.808 | 0.315 |
| word 1grams B.1 | **0.724** | 0.759 | 0.866 | 0.809 | 0.342 |
| word 2grams B.1 | 0.688 | 0.719 | 0.882 | 0.792 | 0.226 |
| word 3grams B.1 | 0.680 | 0.694 | 0.940 | 0.799 | 0.175 |
| LCS B.3 | 0.708 | 0.742 | 0.869 | 0.801 | NaN |
| LCS B.4 | 0.684 | 0.691 | 0.963 | 0.805 | NaN |
| LCSstring B.3 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| LCSstring B.4 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| SKIP2 B.3 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| SKIP2 B.4 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| containment 1gram | 0.721 | 0.779 | 0.819 | 0.798 | 0.350 |
| containment 2gram | 0.677 | 0.754 | 0.775 | 0.764 | 0.270 |
| containment 3gram | 0.656 | 0.739 | 0.757 | 0.748 | 0.226 |
| bigrams | 0.708 | 0.794 | 0.768 | 0.781 | 0.351 |
| trigrams | 0.697 | 0.793 | 0.746 | 0.769 | 0.337 |
| character 2grams | 0.713 | 0.787 | 0.790 | 0.788 | 0.363 |
| character 3grams | 0.705 | 0.788 | 0.769 | 0.779 | 0.347 |
| word Ngrams | 0.679 | 0.786 | 0.721 | 0.752 | **0.385** |
| character Ngrams | 0.706 | 0.802 | 0.750 | 0.775 | 0.360 |
| LCS | 0.704 | 0.756 | 0.831 | 0.791 | NaN |
| LCSstring | 0.678 | 0.679 | 0.993 | 0.806 | 0.058 |
| SKIP2 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| containment | 0.671 | 0.793 | 0.693 | 0.740 | 0.292 |
| Word Character Ngrams | 0.697 | 0.806 | 0.726 | 0.764 | 0.351 |
| LCS LCSstring skip2 containment | 0.668 | 0.800 | 0.677 | 0.734 | 0.302 |
| allMeasures | 0.688 | **0.814** | 0.697 | 0.751 | 0.332 |
| $BASE_1$ | 0.665 | 0.665 | 1.000 | 0.799 | |
| $BASE_2$ | 0.69 | 0.724 | 0.863 | 0.788 | |

Table B.7: Evaluation on paraphrase detection, using a Naive Bayes classifier. Stemming has been employed while we use 70% of the dataset as a training sample and 30% as testing

| SVM | | | | | |
|---|---|---|---|---|---|
| measures | accuracy | precision | recall | F-measure | correlation |
| character bigrams B.1 | 0.717 | 0.739 | 0.899 | 0.811 | 0.323 |
| bigrams B.2a | 0.718 | 0.731 | 0.924 | 0.816 | 0.331 |
| bigrams B.2b | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| bigrams B.2c | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| bigrams B.2d | 0.716 | 0.732 | 0.915 | 0.813 | 0.340 |
| character trigrams B.1 | 0.719 | 0.739 | 0.902 | 0.813 | 0.305 |
| trigrams B.2a | 0.714 | 0.726 | 0.927 | 0.814 | 0.286 |
| trigrams B.2b | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| trigrams B.2c | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| trigrams B.2d | 0.714 | 0.728 | 0.919 | 0.813 | 0.305 |
| word 1grams B.1 | 0.729 | 0.755 | 0.888 | 0.816 | 0.349 |
| word 2grams B.1 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| word 3grams B.1 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| LCS B.3 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| LCS B.4 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| LCSstring B.3 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| LCSstring B.4 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| SKIP2 B.3 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| SKIP2 B.4 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| containment 1gram | 0.726 | 0.750 | 0.893 | 0.815 | 0.348 |
| containment 2gram | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| containment 3gram | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| bigrams | 0.720 | 0.733 | 0.920 | 0.816 | 0.339 |
| trigrams | 0.711 | 0.719 | 0.939 | 0.814 | 0.273 |
| character 2grams | 0.721 | 0.734 | 0.921 | 0.817 | 0.339 |
| character 3grams | 0.711 | 0.720 | 0.935 | 0.813 | 0.265 |
| word Ngrams | 0.734 | 0.755 | 0.898 | 0.820 | 0.343 |
| character Ngrams | 0.719 | 0.728 | 0.933 | 0.818 | 0.329 |
| LCS | 0.697 | 0.706 | 0.944 | 0.808 | NaN |
| LCSstring | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| SKIP2 | 0.675 | 0.675 | 1.000 | 0.806 | NaN |
| containment | 0.736 | 0.757 | 0.898 | 0.821 | 0.353 |
| Word Character Ngrams | 0.742 | 0.761 | 0.900 | 0.825 | 0.384 |
| LCS LCSstring skip2 containment | 0.743 | 0.764 | 0.896 | 0.825 | 0.372 |
| allMeasures | **0.750** | **0.773** | 0.890 | 0.828 | **0.384** |
| $BASE_1$ | 0.665 | 0.665 | 1.000 | 0.799 | |
| $BASE_2$ | 0.69 | 0.724 | 0.863 | 0.788 | |

Table B.8: Evaluation on paraphrase detection, using a SVM classifier. Stemming has been employed while we use 70% of the dataset as a training sample and 30% as testing

## Semantic Textual Similarity

| | Correlation | |
|---|---|---|
| measures | Pearson | Spearman |
| character 2grams B.1 | 0.636 | 0.609 |
| character 2grams B.2a | 0.608 | 0.610 |
| character 2grams B.2b | 0.168 | 0.218 |
| character 2grams B.2c | 0.156 | 0.219 |
| character 2grams B.2d | 0.599 | 0.613 |
| character 3grams B.1 | **0.647** | 0.609 |
| character 3grams B.2a | 0.597 | 0.577 |
| character 3grams B.2b | 0.183 | 0.255 |
| character 3grams B.2c | 0.214 | 0.288 |
| character 3grams B.2d | 0.537 | **0.618** |
| word 1grams B.1 | 0.463 | 0.471 |
| word 2grams B.1 | 0.315 | 0.290 |
| word 3grams B.1 | 0.247 | 0.222 |
| LCS B.3 | 0.319 | 0.342 |
| LCS B.4 | 0.127 | 0.178 |
| LCSstring B.3 | 0.254 | 0.249 |
| LCSstring B.4 | 0.106 | 0.129 |
| SKIP2 B.3 | 0.0036 | 0.095 |
| SKIP2 B.4 | 0.0069 | -0.0147 |
| containment 1gram B.5 | 0.461 | 0.465 |
| containment 1gram B.6 | 0.428 | 0.438 |
| containment 2gram B.5 | 0.346 | 0.330 |
| containment 2gram B.6 | 0.322 | 0.310 |
| containment 3gram B.5 | 0.279 | 0.264 |
| containment 3gram B.6 | 0.264 | 0.254 |

Table B.9: Evaluation on Semantic Textual Similarity by calculating Pearson and Spearman Correlation on the concatenation of Test Datasets of SemEval 2012

| | Correlation | |
|---|---|---|
| measures | Pearson | Spearman |
| character 2grams B.1 | 0.637 | 0.648 |
| character 2grams B.2a | 0.627 | 0.647 |
| character 2grams B.2b | 0.399 | 0.402 |
| character 2grams B.2c | 0.338 | 0.402 |
| character 2grams B.2d | 0.640 | 0.651 |
| character 3grams B.1 | **0.645** | 0.654 |
| character 3grams B.2a | 0.549 | 0.600 |
| character 3grams B.2b | 0.312 | 0.404 |
| character 3grams B.2c | 0.364 | 0.459 |
| character 3grams B.2d | 0.628 | **0.663** |
| word 1grams B.1 | 0.518 | 0.540 |
| word 2grams B.1 | 0.399 | 0.422 |
| word 3grams B.1 | 0.314 | 0.343 |
| LCS B.3 | 0.465 | 0.494 |
| LCS B.4 | 0.271 | 0.359 |
| LCSstring B.3 | 0.386 | 0.409 |
| LCSstring B.4 | 0.239 | 0.317 |
| SKIP2 B.3 | 0.088 | 0.344 |
| SKIP2 B.4 | 0.038 | 0.054 |
| containment 1gram B.5 | 0.501 | 0.512 |
| containment 1gram B.6 | 0.467 | 0.487 |
| containment 2gram B.5 | 0.418 | 0.451 |
| containment 2gram B.6 | 0.380 | 0.419 |
| containment 3gram B.5 | 0.336 | 0.378 |
| containment 3gram B.6 | 0.306 | 0.360 |

Table B.10: Evaluation on Semantic Textual Similarity by calculating Mean Pearson and Spearman on Test Datasets of SemEval 2012

| | Correlation | |
|---|---|---|
| measures | Pearson | Spearman |
| character 2grams B.1 | **0.543** | 0.539 |
| character 2grams B.2a | 0.524 | **0.542** |
| character 2grams B.2b | -0.036 | -0.059 |
| character 2grams B.2c | 0.029 | -0.061 |
| character 2grams B.2d | 0.471 | 0.490 |
| character 3grams B.1 | 0.470 | 0.450 |
| character 3grams B.2a | 0.397 | 0.415 |
| character 3grams B.2b | -0.001 | -0.024 |
| character 3grams B.2c | 0.059 | 0.012 |
| character 3grams B.2d | 0.386 | 0.403 |
| word 1grams B.1 | 0.408 | 0.398 |
| word 2grams B.1 | 0.313 | 0.315 |
| word 3grams B.1 | 0.285 | 0.301 |
| LCS B.3 | 0.272 | 0.263 |
| LCS B.4 | -0.004 | -0.062 |
| LCSstring B.3 | 0.124 | 0.068 |
| LCSstring B.4 | -0.021 | -0.099 |
| SKIP2 B.3 | 0.116 | 0.505 |
| SKIP2 B.4 | -0.048 | -0.043 |
| containment 1gram B.5 | 0.432 | 0.423 |
| containment 1gram B.6 | 0.369 | 0.363 |
| containment 2gram B.5 | 0.334 | 0.342 |
| containment 2gram B.6 | 0.314 | 0.319 |
| containment 3gram B.5 | 0.300 | 0.321 |
| containment 3gram B.6 | 0.291 | 0.313 |

Table B.11: Evaluation on Semantic Textual Similarity by calculating Pearson and Spearman Correlation on the concatenation of Test Datasets of SemEval 2013

| | Correlation | |
|---|---|---|
| measures | Pearson | Spearman |
| character 2grams B.1 | 0.517 | 0.521 |
| character 2grams B.2a | 0.494 | **0.524** |
| character 2grams B.2b | 0.318 | 0.357 |
| character 2grams B.2c | 0.284 | 0.354 |
| character 2grams B.2d | 0.455 | 0.490 |
| character 3grams B.1 | **0.520** | 0.521 |
| character 3grams B.2a | 0.471 | 0.513 |
| character 3grams B.2b | 0.304 | 0.376 |
| character 3grams B.2c | 0.321 | 0.383 |
| character 3grams B.2d | 0.415 | 0.499 |
| word 1grams B.1 | 0.465 | 0.482 |
| word 2grams B.1 | 0.396 | 0.399 |
| word 3grams B.1 | 0.350 | 0.340 |
| LCS B.3 | 0.367 | 0.375 |
| LCS B.4 | 0.233 | 0.296 |
| LCSstring B.3 | 0.352 | 0.330 |
| LCSstring B.4 | 0.246 | 0.272 |
| SKIP2 B.3 | 0.064 | 0.307 |
| SKIP2 B.4 | -0.001 | 0.029 |
| containment 1gram B.5 | 0.472 | 0.484 |
| containment 1gram B.6 | 0.448 | 0.462 |
| containment 2gram B.5 | 0.412 | 0.419 |
| containment 2gram B.6 | 0.404 | 0.406 |
| containment 3gram B.5 | 0.364 | 0.359 |
| containment 3gram B.6 | 0.362 | 0.355 |

Table B.12: Evaluation on Semantic Textual Similarity by calculating Mean Pearson and Spearman Correlation on Test Datasets of SemEval 2013

# Appendix C

# Results for Paraphrasing Prompts

## C.1 Crowsourcing data

The tables below show results for the recognition task among data collected from the CrowdFlower crowdsourcing servise.

| Input prompt | WHAT CITY DO YOU WANT TO FLY TO | |
|---|---|---|
| | without normalization | with normalization |
| Best Candidate Paraphrases | what time do you want to fly<br>where do you want to fly to<br>what do you want to do<br>when do you want to fly<br>where do you want to fly<br>where would you like to fly to<br>where do you like to fly<br>what city did you want to depart from<br>what date do you want to depart<br>what date do you want to travel | where do you want to fly to<br>what time do you want to fly<br>where would you like to fly to<br>what city did you want to depart from<br>where do you want to fly<br>when do you want to fly<br>what do you want to do<br>where do you want to fly to and from<br>where do you want to fly from and to<br>where would you like to fly to please |

Table C.1: Candidate paraphrases for the input prompt: WHAT CITY DO YOU WANT TO FLY TO

| Input prompt | WHERE WOULD YOU LIKE TO GO | |
|---|---|---|
| | without normalization | with normalization |
| Best Candidate Paraphrases | where would you like to go<br>where would you like to start<br>where do you want to go<br>where would you like to start from<br>when would you like to go<br>where and when would you like to go<br>where would you like to arrive<br>where would you like to depart<br>where would you like to fly<br>where would you like to travel | where would you like to go<br>where would you like to start<br>where do you want to go<br>where would you like to start from<br>where and when would you like to go<br>where do you want to go to<br>where do you want to go sir<br>hello where do you want to go<br>where would you like to travel<br>where would you like to fly |

Table C.2: Candidate paraphrases for the input prompt: WHERE WOULD YOU LIKE TO GO

| Input prompt | WHAT TIME DO YOU NEED TO DEPART | |
|---|---|---|
| | without normalization | with normalization |
| Best Candidate Paraphrases | what day do you want to depart<br>what date do you want to depart<br>what time are you looking to depart<br>when do you need to depart<br>what day do you want to leave<br>when do you need to arrive<br>when do you want to depart<br>when do you need to leave<br>what day are you looking to leave<br>what time do you want to fly | what day do you want to depart<br>what date do you want to depart<br>what time are you looking to depart<br>when do you need to depart<br>what day do you want to leave<br>when do you need to arrive<br>when do you want to depart<br>when do you need to leave<br>what day are you looking to leave<br>what time do you want to fly |

Table C.3: Candidate paraphrases for the input prompt: WHAT TIME DO YOU NEED TO DEPART

For the generation task, and the same input prompts, as they are listed at the tables above, the candidate paraphrases are:

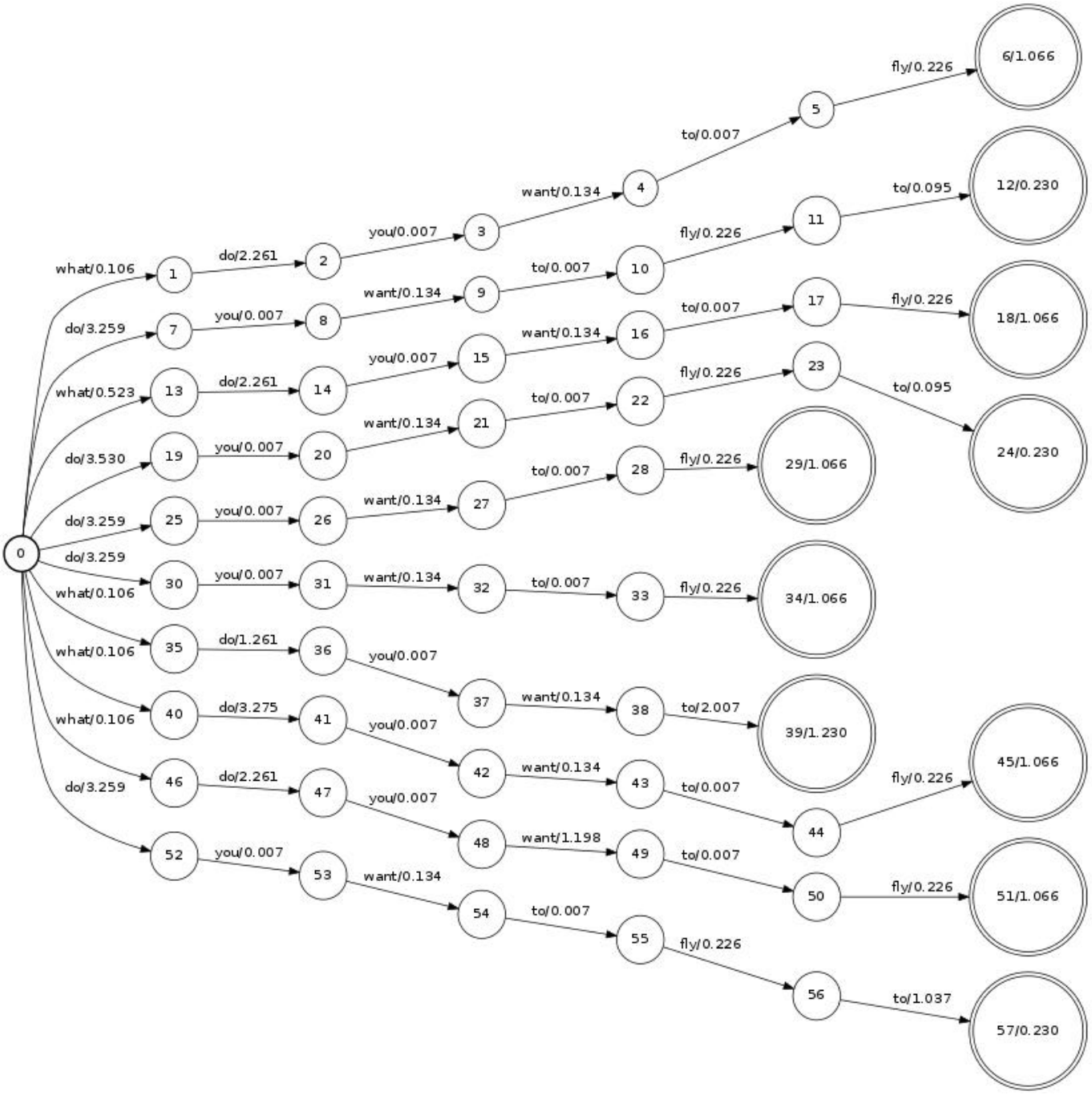



Figure C.1: Candidate Paraphrases for input prompt “WHAT CITY DO YOU WANT TO FLY TO”

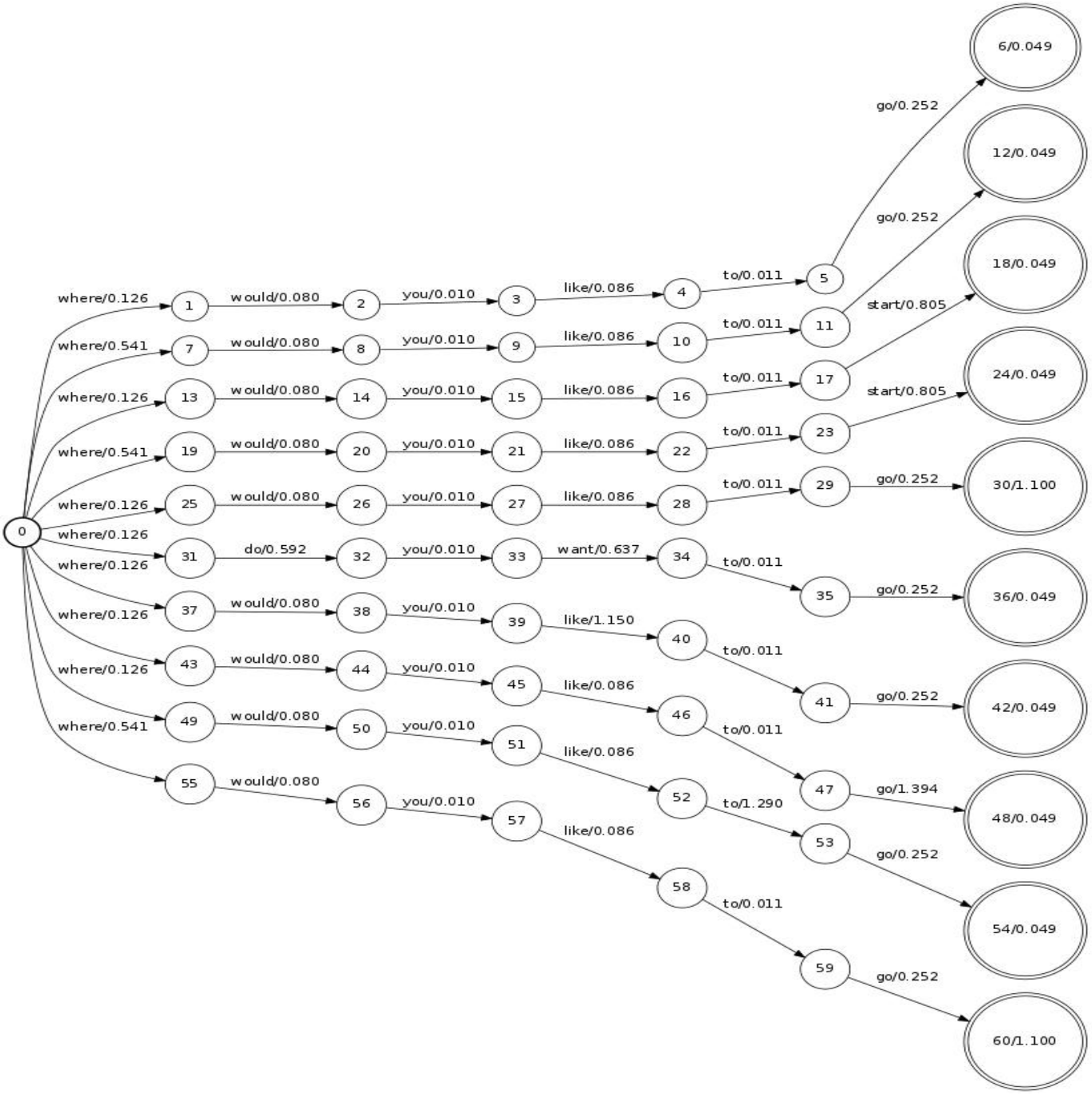



Figure C.2: Candidate Paraphrases for input prompt "WHERE WOULD YOU LIKE TO GO"

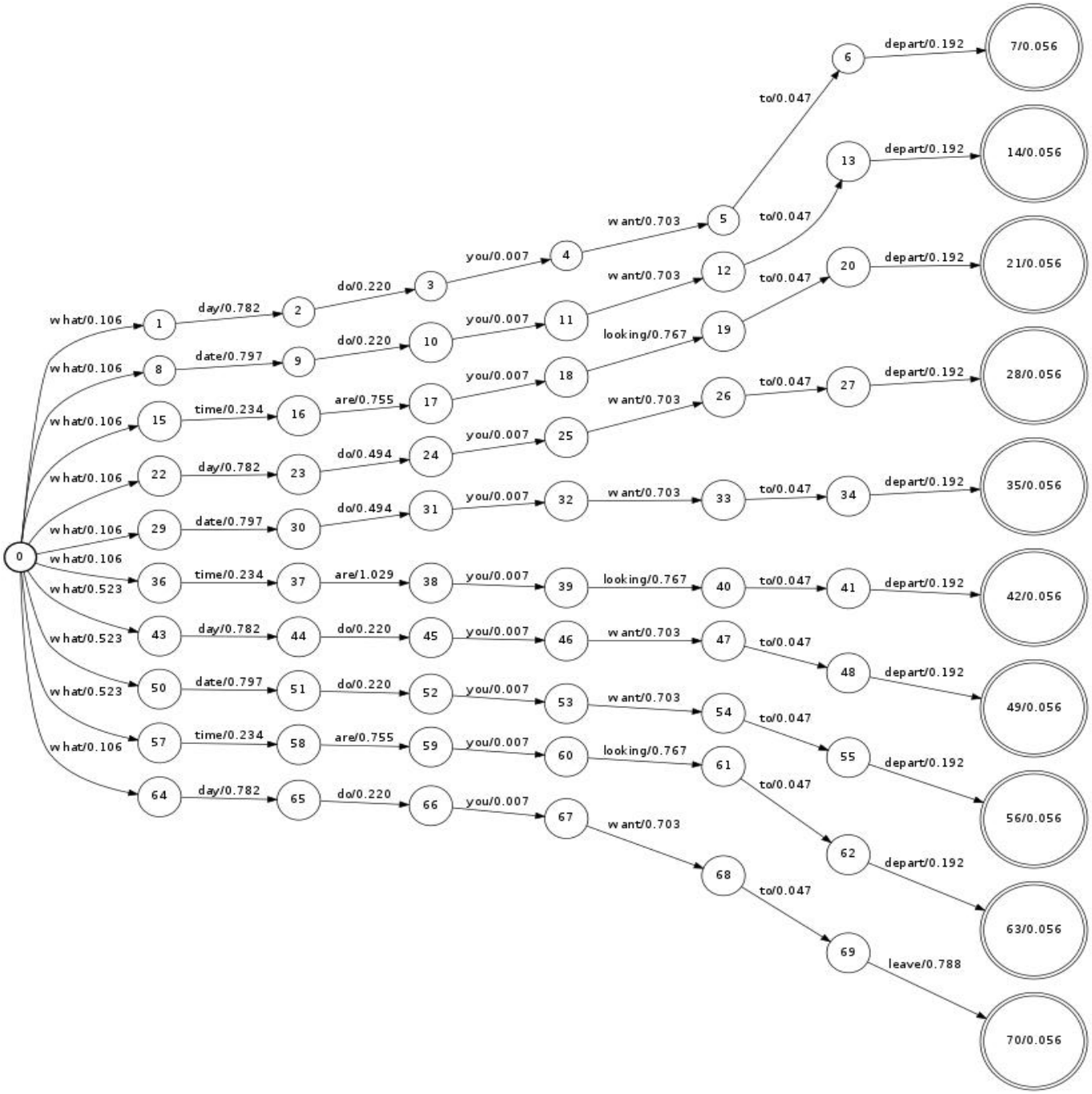



Figure C.3: Candidate Paraphrases for input prompt "WHAT TIME DO YOU NEED TO DEPART"

## C.2 Web documents

The tables below show results for the recognition task among data harvested from the web.

| Category | AIRPORT | |
|---|---|---|
| Input prompt | DO YOU KNOW WHAT AIRPORT | |
| | without normalization | with normalization |
| Best Candidate Paraphrases | do you know what airport<br>do you know what airport it is<br>do you know what airport this is<br>do you know what airport code is aca<br>do you know what airport i m at<br>do you know what airport runway 9 signifies<br>do you know what airport you arrived at<br>do you know what airport hes flying out of<br>do you know what airport inky traveled through today<br>do you know what airport this thai is struggling | do you know what airport<br>do you know what airport this is<br>do you know what airport it is<br>do you know what airport you arrived at<br>do you know what airport runway 9 signifies<br>do you know what airport i m at<br>do you know what airport code is aca<br>do you know what airport you would fly into<br>do you know what airport we re going from<br>do you know what airport this thai is struggling |

Table C.4: Candidate paraphrases for the input prompt: DO YOU KNOW WHAT AIRPORT

| Category | CAR | |
|---|---|---|
| Input prompt | WILL YOU NEED A CAR | |
| | without normalization | with normalization |
| Best Candidate Paraphrases | do you prefer a car service<br>do you prefer a car or truck<br>do you know what a persimmon is<br>what car do you drive<br>what do you look like<br>do you prefer a car buffer or standard<br>do you prefer a car or a house<br>where can i rent a car<br>do you know what tofu is<br>what do you do together | do you prefer a car service<br>do you prefer a car or truck<br>do you know what a persimmon is<br>do you prefer a car or a house<br>do you prefer a car buffer or standard<br>do you prefer a car allowance or have a rental car<br>do you prefer a car service versus a taxi<br>do you prefer a car over any other transportation<br>do you prefer a car like chassis with better handling<br>are you looking to buy a new car or a quality used car |

Table C.5: Candidate paraphrases for the input prompt: WILL YOU NEED A CAR

| Category | DATE | |
|---|---|---|
| Input prompt | WHAT DATE WOULD YOU LIKE TO TRAVEL | |
| | without normalization | with normalization |
| Best Candidate Paraphrases | what other date would you like<br>what would you like to ask<br>what name do you like best<br>what digimon do you like<br>what do you like better<br>what would you like to see more of<br>what gun would you prefer<br>what would you like to see in colusa county<br>which date would you prefer<br>which date would you prefer to attend | what would you like to ask<br>what other date would you like<br>what name do you like best<br>what would you like to see more of<br>what would you like to see in colusa county<br>what do you like better<br>what digimon do you like<br>what did you do to your mini today<br>what do you want your baby to be<br>what weekend would you like to have the roosevelt reunion |

Table C.6: Candidate paraphrases for the input prompt: WHAT DATE WOULD YOU LIKE TO TRAVEL

| Category | DAY | |
|---|---|---|
| Input prompt | WHAT DAY WOULD YOU LIKE TO DEPART | |
| | without normalization | with normalization |
| Best Candidate Paraphrases | what day would you want to come<br>what day would you want to live<br>what day would you want to repeat<br>what would you like to ask<br>what day would you want to go<br>what day would you want to be born<br>what day would you want to do training<br>what day would you want to meet up<br>what day would you want to never end<br>what day would you want to pick up | what day would you want to repeat<br>what day would you want to live<br>what day would you want to come<br>what would you like to ask<br>what day would you want to relive<br>what day would you want to pick up<br>what day would you want to never end<br>what day would you want to meet up<br>what day would you want to do training<br>what day would you want to be born |

Table C.7: Candidate paraphrases for the input prompt: WHAT DAY WOULD YOU LIKE TO DEPART

| Category | PLACE | |
|---|---|---|
| Input prompt | WHAT CITY DO YOU WANT TO FLY TO | |
| | without normalization | with normalization |
| Best Candidate Paraphrases | what do you want to do<br>what do you want to see<br>why do you want to fly to detroit<br>what city do you live in<br>what do you drunks want<br>what do you guys want to see<br>what do you want him to explain<br>what do you want me to do<br>what would you like to ask<br>what are you going to do | why do you want to fly to detroit<br>what do you want to see<br>what do you want to do<br>what city do you live in<br>what do you want me to do<br>what do you want him to explain<br>what do you guys want to see<br>what would you like to ask<br>what are you going to do<br>what do you do to keep fit |

Table C.8: Candidate paraphrases for the input prompt: WHAT CITY DO YOU WANT TO FLY TO

| Category | TIME | |
|---|---|---|
| Input prompt | WHAT TIME DO YOU NEED TO DEPART | |
| | without normalization | with normalization |
| Best Candidate Paraphrases | what time would you want to leave<br>what time would you want to go<br>what time would you want to know<br>what time would you want to visit<br>what do you want to be<br>what time do you have to eat dinner<br>at what time would you want to return<br>what time would you want to eat dinner<br>what time would you want to get going<br>what time would you want to get there | what time would you want to leave<br>what time would you want to visit<br>what time would you want to know<br>what time would you want to go<br>what time do you have to eat dinner<br>what time would you want to arrive at the falls<br>what time would you want to arrive at and why<br>what time would you want to meet there<br>what time would you want to live in<br>what time would you want to go go |

Table C.9: Candidate paraphrases for the input prompt: WHAT TIME DO YOU NEED TO DEPART

# Bibliography


[1] Aggarwal, N., Asooja, K. and Buitelaar, P., "*DERI&UPM: Pushing Corpus Based Relatedness to Similarity: Shared Task System Description.*" Proc. of the 1st Joint Conference on Lexical and Computational Semantics (*Sem), pp. 643-647, Montreal, Canada, 2012.

[2] Androutsopoulos, I. and Malakasiotis, P., "*A survey of Paraphrasing and Textual Entailment Methods.*" Journal of Artifial Intelligence Research 38(1), pp. 135-187, 2010.

[3] Banea, C., Hassan, S., Mohler, M. and Mihalcea, R., "*UNT: A Supervised Synergistic Approach to Semantic Text Similarity.*" Proc. of the 1st Joint Conference on Lexical and Computational Semantics (*Sem), pp. 635-642, Montreal, Canada, 2012.

[4] Bannard, C. and Allison-Burch, C., "*Paraphrasing with bilingual parallel corpora.*" Proc. of the 43rd Annual Meeting of ACL, pp. 597-604, Ann Arbor, USA, 2005.

[5] Bar, D., Biemann, C., Gurevych, I. and Zesch, T., "*UKP: Computing Semantic Textual Similarity by Combining Multiple Content Similarity Measures.*" Proc. of the 1st Joint Conference on Lexical and Computational Semantics (*Sem), pp. 435-440, Montreal, Canada, 2012.

[6] Bar-Haim, R., Degan, I., Greental, I. and Shnarch, E., "*Semantic inference at the lexical-syntactic level.*" Proc. of the 22nd Conf. on Artificial Intelligence, pp. 871-876, Vancouver, Canada, 2007.

[7] Barzilay, R. and Lee, L., "*Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment.*" Proc. of the HLT Conf. of NAACL, pp. 16-23, Edmonton, Canada, 2003.

[8] Bateman, J. and Zock, M, "*Natural Language Generation.*" In Mitkov, R. (Ed.), The Oxford Handbook of Comp. Linguistics, chap. 15, pp. 284-304. Oxford University Press, 2003.

[9] Bhagat, R., Pantel, P. and Hovy, E., "*LEDIR: An unsupervised algorithm for learning directionality of inference rules.*" Proc. of the Conf. on EMNLP and the Conf. on Computational Nat. Lang. Learning, pp. 161-170, Prague, Czech Republic, 2007.

[10] Bhagwani, S., Satapathy, S. and Karnick, H., "*sranjans: Semantic Textual Similarity using Maximal Weighted Bipartite Graph Matching.*" Proc. of the 1st Joint Conference on Lexical and Computational Semantics (*Sem), pp. 579-585, Montreal, Canada, 2012.

[11] Biggins, S., Mohammed, S. and Oakley, S., "*University_of_Sheffield: Two Approaches to Semantic Textual Similarity.*" Proc. of the 1st Joint Conference on Lexical and Computational Semantics (*Sem), pp. 435-440, Montreal, Canada, 2012.

[12] Bos, J. and Market, K., "*Recognising textual entailment with logical inference.*" Proc. of the Conf. on HLT and EMNLP, pp. 628-635, Vancouver, Canada, 2005.

[13] Brazilay, R. and Lee, L., "*Bootstrapping lexical choice via multiple-sequence alignment.*" Proc. of the Conf. on EMNLP, pp. 164-171, Philadelphia, USA, 2002.

[14] Brazilay, R. and Lee, L., "*Learning to paraphrase: an unsupervised approach using multiple-sequence alignment.*" Proc. of the HLT Conf. of NAACL, pp. 16-23, Edmonton, Canada, 2003.

[15] Brazilay, R. and McKewon, K., "*Extracting paraphrases from a parallel corpus.*" Proc. of the 39th Annual Meeting of ACL, pp. 50-57, Toulouse, France, 2001.

[16] Brockett, C. and Dolan, W., "*Support Vector Machines for paraphrase identification and corpus construction.*" Proc. of the 3rd Int. Workshop on Paraphrasing, pp. 1-8, Jeju island, Korea, 2005.

[17] Buscaldi, D., Tournier, R., Aussenac-Gilles, N. and Mothe, J., "*IRIT: Textual Similarity Combining Conceptual Similarity with an N-Gram Comparison Method.*" Proc. of the 1st Joint Conference on Lexical and Computational Semantics (*Sem), pp. 552-556, Montreal, Canada, 2012.

[18] Califf, M. and Mooney, R., "*Bottom-up relational learning of pattern matching rules for information extraction.*" Journal of Machine Learning Research, 4(Jun), pp. 177-210, 2003.

[19] Caputo, A., Basile, P. and Semeraro, G., "*UNIBA: Distributional Semantics for Textual Similarity.*" Proc. of the 1st Joint Conference on Lexical and Computational Semantics (*Sem), pp. 591-596, Montreal, Canada, 2012.

[20] Castilo, J. and Estrella, P., "*SAGAN: An approach to Semantic Textual Similarity based on Textual Entailment.*" Proc. of the 1st Joint Conference on Lexical and Computational Semantics (*Sem), pp. 667-672, Montreal, Canada, 2012.

[21] Cohn, T. and Lapata, M., "*Sentence Compression beyond word deletion.*" Proc. of the 22nd Int. Conf. on Comp. Linguistics, Manchester, UK, 2008.

[22] Croce, D., Annesi, P., Storch, V. and Basili, R., "*Unitor: Combining Semantic Text Similarity functions through SV Regression.*" Proc. of the 1st Joint Conference on Lexical and Computational Semantics (*Sem), pp. 597-602, Montreal, Canada, 2012.

[23] Duboue, P.A., and Chu-Carroll, J., "*Answering the question you wish they had asked: The impact of paraphrasing for question answering.*" Proc. of the HLT Conf. of NAACL, pp. 33-36, New York, USA, 2006.

[24] Fellbaum, C., "*WordNet: An Electronic Lexical Database.*" MIT Press, 1998.

[25] Galanis, D. and Androutsopoulos, I., "*An extractive supervised two-stage method for sentence compression.*" Proc. of the HLT Conf. of NAACL, Los Angeles, USA, 2010.

[26] Grishman, R., "*Information Extraction.*" In Mitkov, R. (Ed.), The Oxford Handbook of Comp. Linguistics, chap. 30, pp. 545-559. Oxford University Press, 2003.

[27] Haghighi, A. D., "*Pobust textual inference via graph matching.*" Proc. of the HLT Conf. on EMNLP, pp. 387-394, Vancouver, Canada, 2005.

[28] Harabagiu, S. and Hickl, A., "*Methods for using textual entailment in open-domain question answering.*" Proc. of the 21st National Conf. on Artifial Intelligence, pp. 755-762, Boston, USA, 2006.

[29] Harmeling, S., "*Inferring textual entailment with a probabilistically sound calculus.*" Nat. Lang. Engineering, 15(4), pp. 459-477, 2009.

[30] Harris, Z., "*Distributional Structure.*" In Katz, J. and Fodor, J. (Eds.), The Philosphy of Linguistics, pp. 33-49. Oxford University Press, 1964.

[31] Hovy, E., "*Text Summarization.*" Mitkov, R. (Ed.), The Oxford Handbook of Comp.Linguistics, chap.32, pp. 583-598. Oxford University Press, 2006.

[32] Huffman, S., "*Learning information extraction patterns from examples.*" Proc. of the IJCAI Workshop on New Approaches to Learning for Nat. Lang. Processing, pp. 127-142, Montreal, Canada, 1995.

[33] Ibrahim, A., Katz, B. and Lin, J., "*Extracting Structural paraphrases from aligned monolingual corpora.*" Proc. of the ACL Workshop on Paraphrasing, pp. 57-64, Sapporo, Japan, 2003.

[34] Iftene, A. and Balahur-Dobrescu, A., "*Hypothesis trasformation and semantic variability rules used in recognizing textual entailment.*" Proc. of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing, pp. 125-130, Prague, Czech Republic, 2007.

[35] Iosif, E. and Potamianos, A., "*Unsupervised Semantic Similarity Computation Between Terms Using Web Documents.*" IEEE Transactions on Knowledge and Data Engineering, 22(11), 2009.

[36] Jimenez, S., Becerra, C. and Gelbukh, A., "*Soft Cardinality: A Parameterized Similarity Function for Text Comparison.*" Proc. of the 1st Joint Conference on Lexical and Computational Semantics (*Sem), pp. 449-453, Montreal, Canada, 2012.

[37] Knight, K., and Marcu, D., "*Summarization beyond sentence extraction: A probalistic approach to sentence compression.*" Artificial Intelligence, 139(1), pp. 91-107, 2002.

[38] Koehn, P., "*Statistical Machine Translation.*" Cambridge University Press, 2009.

[39] Koehn, P., Och, F.J. and Marcu, D., "*Statistical phrase-based translation.*" Proc. of the HLT Conf. of NAACL, pp. 48-54, Edmonton, Canada. ACL, 2003.

[40] Levenshtein, V., "*Binary codes capable of correcting deletions, insertions and reversals.*" Soviet Physice-Doklady, 10(8), pp. 707-710, 1966.

[41] Lin, D., "*An information-theoretic definition of similarity.*" Proc. of the 15th Int. Conf. on Machine Learning, pp. 296-304, Madison, WI. Morgan Kaufmann, San Francisco, USA, 1998.

[42] Lin, D., "*Automatic retrieval and clustering of similar words.*" Proc. of the 36th Annual Meeting of ACL and 17th Int. Conf. on Comp. Linguistics, pp. 768-774, Montreal, Canada, 1998.

[43] Lin, D. and Pantel, P., "*Discovery of inference rules for question answering.*" Nat. Lang. Engineering, 15(3), pp. 414-453, 2001.

[44] Malakasiotis, P. and Androutsopoulos, I., "*Learning textual entailment using SVMs and string similarity measures.*" Proc. of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing, pp. 42-47, Prague. ACL, 2007.

[45] Malandrakis, N., Iosif, E. and Potamianos, A., "*Deep Purple: Estimating Sentence Semantic Similarity using N-gram Regression Models and Web Snippets.*" Proc. of the 1st Joint Conference on Lexical and Computational Semantics (*Sem), pp. 565-570, Montreal, Canada, 2012.

[46] Malandrakis, N., Iosif, E., Prokopi, V., Potamianos, A. and Narayanan, S., "*Deep-Purple: Lexical, String and Affective Feature Fusion for Sentence-Level Semantic Similarity Estimation.*" Proc. of the 2nd Joint Conference on Lexical and Computational Semantics (*Sem), Volume 1: Proceedings of the Main Conference and the shared Task, pp. 103-108, Atlanta, USA, 2013.

[47] Mani, I., "*Automatic Summarization.*" John Benjamins, 2001.

[48] Manning, C.D. and Schuetze, H., "*Foudations of Statistical Natural Language Processing.*" MIT Press, 1999.

[49] Marquez, L. and Carreras, X., Litkowski, K. C. and Stevenson, S., "*Semantic role labeling: an introduction to the specia issue.*" Comp. Linguistics, 34(2), pp. 145-159, 2008.

[50] McDonald, R, "*Discriminative sentence compression with soft syntactic constraints.*" Proc. of the 11th Conf. of EACL, pp. 297-304, Trento, Italy, 2006.

[51] McKeown, K., "*Paraphrasing questions using given and new information.*" Comp. Linguistics, 9(1), 1983.

[52] Moens, M., "*Information Extraction: Algorithms and Prospects in a Retrieval Context.*" Springer, 2006.

[53] Moore, R. C., "*Towards a simple and accurate statistical approach to learning translation relationships among words.*" Proc. of the ACL Workshop on Data-Driven Machine Translation, Toulouse, France, 2001.

[54] Neogi, S., Pakray, P., Bandyopadhyay, S. and Gelbukh, A., "*JU_CSE_NLP: Multi-grade Classification of Semantic Similarity Between Text Pairs.*" Proc. of the 1st Joint Conference on Lexical and Computational Semantics (*Sem), pp. 571-574, Montreal, Canada, 2012.

[55] Nitin Madnani and Bonnie J. Dorr, "*Generating Phrasal and Sentential Paraphrases: A survey of Data-Driven Methods.*" Journal of Computational Linguistics, 36(3), pp. 341-387, MIT Press Cambridge, USA, 2010.

[56] Palogiannidi, E., "*Using crowdsoursing for grammar induction with application to spoken dialogue systems.*" Diploma Thesis, Technical University of Crete, Chania, Greece, 2013.

[57] Pang, B., Knight, K. and Marcu, D., "*Syntax-based alignment of multiple translations: extracting paraphrases and generating new sentences.*" Proc. of the Human Lang. Techn. Conf. of NAACL, pp. 102-109, Edmonton, Canada, 2003.

[58] Quirk, C., Brockett, C. and Dollan, W.B., "*Monolingual machine translation for paraphrase generation.*" Proc. of the Conf. on EMNLP, pp. 142-149, Barcelona, Spain, 2004.

[59] Riezler, S., Vasserman, A., Tsochantaridis, I., Mittal, V., and Liu, Y., "*Statistical machine translation for qury expansion in answer retrieval.*" Proc. of the 45th Annual Meeting of ACL, pp. 464-471, Prague, Czech Republic, 2007.

[60] Riloff, E. and Jones, R., "*Learning dictionaries for information extraction by multi-level bootstrapping.*" Proc. of the 16th National Conf. on Artificial Intelligence, pp. 474-479, Orlando, USA, 1999.

[61] Schohn, G. and Cohn, D., "*Less in more: active learning with Support Vector Machines.*" Proc. of the 17th Int. Conf. on Machine Learning, pp. 839-846, Stanford, USA, 2000.

[62] Selkow, S., "*The tree-to-tree editing problem.*" Information Processing Letters, 6(6), pp. 184-186, 1977.

[63] Shinyama, Y. and Sekine, S., "*Paraphrase acquisition for information extraction.*" Proc. of the ACL Workshop on Paraphrasing, Sapporo, Japan, 2003.

[64] Singh, J., Bhattacharya, A. and Bhattacharya, P., "*janardhan: Semantic Textual Similarity using Universal Networking Language graph matching.*" Proc. of the 1st Joint Conference on Lexical and Computational Semantics (*Sem), pp. 662-666, Montreal, Canada, 2012.

[65] Soderland, S., Fisher, D., Aseltine, J. and Lehnert, W.G., "*CRYSTAL: Inducing a conceptual dictionary.*" Proc. of the 14th Int. Joint Conf. on Artificial Intelligence, pp. 1314-1319, Montreal, Canada, 1995.

[66] Souza, J.G.C., Negri, M. and Mehdad, Y., "*FBK: Machine Translation Evaluation and Word Similarity metrics for Semantic Textual Similarity.*" Proc. of the 1st Joint Conference on Lexical and Computational Semantics (*Sem), pp. 624-630, Montreal, Canada, 2012.

[67] Szpektor, I., Tanev, H., Dagan, I. and Coppola, B., "*Scaling Web-based acquisition of entailment relations.*" Proc. of the Conf. on EMNLP, Barcelona, Spain, 2004.

[68] Tai, K.-C., "*The tree-to-tree correction problem.*" Journal of ACM, 26(3), pp. 422-433, 1977.

[69] Tatu, M. and Moldovan, D., "*A semantic approach to recognizing textual entailment.*" Proc. of the Conf. on HLT and EMNLP, pp. 371-378, Vancouver, Canada, 2005.

[70] Tatu, M. and Moldovan, D., "*COGEX at RTE 3.*" Proc. of the ACL-PASKAL Workshop on Textual Entailment and Paraphrasing, pp. 33-40, Prague, Czech Republic, 2007.

[71] Tong, S. and Koller, D., "*Support Vector Machine active learning with applications to text classification.*" Journal of Machine Learning Research, 2(Nov), pp. 45-66, 2002.

[72] Wang, M. and Cer, D., "*Stanford: Probabilistic Edit Distance Metrics for STS.*" Proc. of the 1st Joint Conference on Lexical and Computational Semantics (*Sem), pp. 648-654, Montreal, Canada, 2012.

[73] Xu, F., Uszkoreit, H. and Li, H., "*A seed-driven bottom-up machine learning framework for extracting relations of various complexity.*" Proc. of the 45th Annual Meetting of the Association of Comp. Linguistics, pp. 584-591, Prague, Czech Republic, 2007.

[74] Zhang, K. and Shasha, D., "*Simple fast algorithms for editing distance between trees and related problems.*" SIAM Journal of Computing, 18(6), pp. 1245-1262, 1989.

[75] Zhao, S., Lan, X., Liu, T. and Li, S., "*Pivot Application-driven statistical paraphrase generation.*" Proc. of the 47th Annual Meeting of ACL and the 4th Int. Joint Conf. on Nat. Lang. Processing of AFNLP, pp. 834-842, Singapore, 2009.

[76] Zhao, S., Wang, H., Liu, T. and Li, S., "*Pivot Approach for extracting paraphrase patterns from bilingual corpora.*" Proc. of the 46th Annual Meeting of ACL: HLT, pp. 780-788, Columbus, USA, 2008.