University of West Bohemia
Faculty of Applied Sciences
Department of Computer Science and Engineering

Master Thesis

Advanced Methods for Sentence Semantic Similarity

Pilsen, 2012

Tomáš Ptáček

Acknowledgments

I would like to thank to Ing. Ivan Habernal for his guidance, and my parents and friends for their love and support.

I hereby declare that this diploma thesis is completely my own work and that I used only the cited sources.

Pilsen, 15. května 2012

Tomáš Ptáček

Abstract

This thesis is focused on the problem of sentence semantic similarity in English language. The theory for document preprocessing, information retrieval models, semantic methods in information retrieval, semantic similarity between words and sentence similarity measures are introduced in the theoretical part. We selected and implemented five promising sentence similarity measures. In the practical part we propose six new sentence similarity measures inspired by state-of-the-art measures described in the theoretical part. We evaluate eleven sentence similarity measures. The evaluation is conducted on two different data sets. The data sets are the Microsoft Research paraphrase corpus and the Semantic Textual Similarity shared task. At the end of the thesis the results are discussed.

Contents

1 Introduction

1.1 Problem description

Determining the similarity between sentences is one of the crucial tasks in natural language processing (NLP). It has wide impact in many text-related research fields. For example, in text summarization, sentence semantic similarity is used to cluster similar sentences. In web page retrieval, sentence similarity can enhance effectiveness by calculating similarities of page titles. [24] In information retrieval, similarity measure is used to assign a ranking score between a query and texts in a corpus. These are only a few examples of sentence semantic similarity applications.

Computing sentence similarity is not a trivial task, due to the variability of natural language expressions. Techniques for detecting similarity between long texts (documents) focus on analyzing shared words, but in short texts word co-occurrence may be rare or even null. That is why sentence semantic similarities incorporate the syntactic and semantic information that can be extracted at the sentence level. However techniques for detecting similarity between long texts must still be taken into account because their adaptation can be used to compute sentence similarity.

Information retrieval studies similarity between long texts and queries. This is very similar to measuring similarity between short texts (sentences) and the same principles and techniques or their modified versions can be used to solve our problem.

The result of this work is the evaluation of five state-of-the-art sentence similarity measures and six proposed sentence similarity measures. These sentence similarity measures are evaluated on two different data sets.

1.2 Outline

In this work we describe the main information retrieval models (section 2.2) and promising sentence semantic similarity measures (section 2.5). In section 4.2 we present six new sentence similarity measures. Implementation is described in section 5. We evaluate these measures on two different data sets in section 6. The data sets are described in section 2.6. At the end of this thesis in section 7 we suggest future work and summarize contribution of this thesis.

1.3 Assignment

1. Study computer methods for sentence semantic similarity.
2. Study the existing solutions with regard to the algorithms and suitability for solving the given problem.
3. Suggest your own extension, experimentally verify the results achieved on the supplied data.
4. Critically evaluate the work results.

2 Theoretical basis

This section describes basic document preprocessing techniques, information retrieval models, semantic methods in information retrieval, semantic similarity between words and sentence similarity measures. Most of the theory is associated with the information retrieval (IR) field, therefore we will describe its usage.

IR systems are used to find documents (large units of text) that satisfy user query. The query consists of a small number of keywords (tokens). Tokens are further decomposed to terms. Terms are the indexed units (usually words, but numbers and dates as well), they are usually derived from tokens by various normalization processes. The IR system compares given query with the number of occurrences of query terms in individual documents and computes document similarity based on used similarity measure. To avoid scanning texts for each query we can index the documents in advance, this means to build a term-document incidence matrix. Rows of the matrix represent vector for given term, which reveals the documents where the term appears in and number of occurrences. Columns of the matrix show us which terms appear in given document and how many times. [21][22]

2.1 Document preprocessing

First we must obtain tokens from the documents and apply various normalization processes to adjust them for our specific needs. Different languages use specific preprocessing techniques mostly because of grammatical and morphological reasons. The goal of this phase is to reduce inflectional forms of words to a common base form. In this section the basic preprocessing techniques are discussed.

2.1.1 Tokenization

Tokenization is the task of chopping up documents into tokens and throwing away punctuation and other unwanted characters. The same process must be applied to document and query to assure that a sequence of characters in text will match the same sequence typed in the query. [22]

2.1.2 Stop Words

Some words that occur in most documents have a small impact in the selection of documents matching user query. These words are excluded from

the vocabulary and we call them stop words. To determine a list of stop words (stop list) the terms in the document collection are sorted by collection frequency (number of occurrences of terms in document collection), and the most frequent terms with little or none semantic value relative to the domain of the documents are then discarded. Semantic content of documents must be taken into account when selecting the stop words. [22]

2.1.3 Lemmatization

Lemmatization is a technique from Natural Language Processing which does full morphological analysis and identifies the base or dictionary form of a word, which is known as the lemma. [21]

2.1.4 Stemming

Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of retrieving the stem of the word correctly most of the time. It often includes the removal of derivational affixes. [22]

2.2 Information Retrieval Models

There are four main IR models: Boolean model, vector space model, language model and probabilistic model. Most commonly used models in IR systems are the first three.

2.2.1 Boolean Model

The Boolean model is one of the basic information retrieval models. The query and the retrieval are based on Boolean algebra. The exact document representation is used to match documents to the user query. [21]

Document Representation

Documents and queries are represented as sets of terms. Each term in document is only considered present (term weight = 1) or absent (term weight = 0).

Boolean Queries

Boolean operators AND, OR, and NOT, are used to logically combine query terms.

Document Retrieval

The system retrieves every document that makes the query logically true. The retrieval is based on the binary decision criterion. Document may be either relevant or not relevant. This is called exact match. Major disadvantage of the Boolean model is that there is no partial match or ranking of the retrieved documents. The relevance of document is based upon the terms frequency and their proximity. Due to this problem, the Boolean model is rarely used alone in practice. [21]

2.2.2 Vector Space Model

This model is perhaps the most widely used IR model.

Document Representation

A document in the vector space model is represented as a weight vector, in which each component weight is computed based on some variation of Term Frequency (TF) or Term Frequency - Inverse document frequency (TF-IDF) scheme. The weight w_{ij} of term t_i in document d_j is no longer in 0, 1 as in the Boolean model, but it can be any number. [21]

Term Frequency Scheme

In document d_j weight of a term t_i is the number of occurrences of t_i in document d_j, denoted by f_{ij}. This scheme does not take into consideration that one term may appear in many documents of the collection. Such term may not be discriminative. [21]

Term Frequency - Inverse Document Frequency Scheme

This is the best known weighting scheme. This scheme has many variations. Let N be the total number of documents in the collection and d_{fi} be the number of documents in which term t_i appears at least once. Then the term weight is computed according to this formula:

$$w_{iq} = tf_{ij} * \frac{logN}{df_i} \tag{2.1}$$

With the use of TF-IDF scheme the terms that appear in all documents have low weight and unique terms have high weight. [17]

Queries

Since query is used to find the document that corresponds with the query it is represented the same way as a document. [21]

Document Retrieval and Relevance Ranking

Unlike the Boolean model, the vector space model does not make a binary decision on whether a document is relevant to a query. Instead, the documents are ranked according to their degrees of relevance to the query. One way to compute the degree of relevance is to calculate the similarity of the query to each document in the collection. [21]

There are many similarity measures. Since the documents and the query are represented as a vector the easiest way to describe similarity is to compute the angle between these vectors. The de facto standard in this area is the cosine similarity, which is the cosine of the angle between the query vector x and the document vector y. [22]

$$sim(X, Y) = \frac{\sum_{j=1}^{n}(x_j y_j)}{\sqrt{\sum_{j=1}^{n}(x_j)^2 \sum_{j=1}^{n}(y_j)^2}} \tag{2.2}$$

Another similarity measures are for example Jaccard a Czekanowski (Dice) coefficient. These measures are commonly used for asymmetric binary variables. We can describe both with the following formula:

$$sim(X, Y) = \frac{\Theta \sum_{j=1}^{n}(x_j y_j)}{\Theta \sum_{j=1}^{n}(x_j y_j) + \sum_{j=1}^{n} |x_j - y_j|} \tag{2.3}$$

If $\Theta = 1$, than represents Jaccard coefficient and if $\Theta = 2$, than it expresses Dice coefficient. [16]

2.2.3 Statistic Language Model

This approach is based on probability and statistics. First a language model is created for each document and then the documents are evaluated based on query probability for given document language model. Let the query q be a sequence of terms $q = q_1, q_2, \ldots, q_m$ and document collection D is a set of documents $d_1, d_2, \ldots, d_N$, then the probability that document d_j "generates" query q is $Pr(q|d_j)$. For document ranking a posteriori probability $Pr(d_j|q)$ is used. Using the Bayes rule, we get:

$$Pr(d_j|q) = \frac{Pr(q|d_j)Pr(d_j)}{Pr(q)} \tag{2.4}$$

$Pr(q)$ and $Pr(d_j)$ is neglected, because the values don't affect rating. $Pr(q)$ will be the same for all documents and probability $Pr(d_j)$ is considered as uniform. For multi-word query we consider the terms are independent and as a result we get this formula:

$$Pr(q = q_1 q_2 ... q_m | d_j) = \prod_{i=1}^{m} Pr(q_i|d_j) = \prod_{i=1}^{|V|} Pr(t_i|d_j) Pr(d_j)^{f_{iq}}, \tag{2.5}$$

where f_{iq} is the total number of occurrences of term t_i in query q and $\sum_{i=1}^{|V|} Pr(t_i|d_j) = 1$. Problem is reduced to estimating the relative frequency:

$$Pr(t_i|d_j) = \frac{f_{ij}}{|d_j|}, \tag{2.6}$$

where f_{ij} is the number of occurrences of term t_i in document d_j and $|d_j|$ denotes for the total number of terms in document d_j. If a term doesn't appear in document then the probability is zero, which underestimates the probability of given term in document. To avoid exclusion of documents from the results, we use smoothing, which works by slightly increasing the probability of low values and decreasing the probability of high values in attempt to refine the model. For additive smoothing the next formula is being used.

$$Pr_{add}(t_i|d_j) = \frac{\lambda + f_{ij}}{\lambda|V| + |d_j|} \tag{2.7}$$

If λ equals one, then the smoothing is called Laplace smoothing, else if $0 < \lambda < 1$ it's called Lidstone smoothing. [21]

2.3 Semantic Methods in IR

2.3.1 Latent Semantic Analysis

Latent semantic analysis (LSA) is the most commonly used method using vector spaces and dimension reduction for extracting and representing the similarity of meaning of words.

During semantic space LSA language construction, first a matrix is created, where position contains information about the term weight in given document. Because the vocabulary is based on the whole document collection, a single document matrix is quite large and therefore we further reduce its dimension. Commonly used mathematical technique used with the dimension reduction property is called singular value decomposition (SVD). During the

compression part of the information is lost and some similar vectors are united or getting closer in the semantic vector space. [14]

Document similarity is calculated with the use of cosine similarity measure or other vector similarity measures.

LSA has its limits. It doesn't take into consideration the word order in sentence, syntactic relations and morphology. Surprisingly it manages to extract correct text representation even without these indicators, but we must still assume that the results are incomplete or can contain an error. [19]

Singular Value Decomposition

Singular value decomposition computes approximated matrix A_k, where k is the dimension of the matrix. The rank of document matrix $A(m \times n)$ is $r \leq \min(m, n)$. First the document matrix A is decomposed into the product of three matrices

$$A = U\Sigma V^T, \tag{2.8}$$

where $U(m \times r)$ and $V^T(n \times r)$ are orthogonal matrices with eigenvectors of AA^T respective A^TA. $\Sigma(r \times r)$ is a diagonal matrix with singular values (non-negative square roots of eigenvalues of AA^T arranged in decreasing order). The main idea of SVD is to select only k largest singular values in Σ by dropping the last $r - k$ columns in all three matrices. The approximated matrix A_k consists of the reduced matrices.

$$A_k = U_k\Sigma_k V_k^T, \tag{2.9}$$

We also need to transform the query q into the reduced form q_k which is shown in the following equation.

$$q = U_k\Sigma^k q_k^T \tag{2.10}$$

This can be modified to the following form.

$$q_k = q^T U_k \Sigma_k^{-1} \tag{2.11}$$

Retrieval can be done by comparing q_k with rows in V_k (each row corresponds to a document) or comparing $\Sigma^k q_k^T$ with rows in $\Sigma_k V_k^T$. It is not clear which method has better results. [21]

2.3.2 Hyperspace Analogue to Language

Hyperspace Analogue to Language (HAL) creates semantic space from word co-occurrences. The analysis is performed by placing the examined word at

the beginning of a moving window of length n which records the relation between adjacent words. The weighted number of occurrences is written in a matrix. Words closer to the examined word have higher weight because they may contain more semantic information associated with the analyzed word. HAL records the information about the word order by distinguishing whether the adjacent word is before or after the examined word. For further information see [9].

2.3.3 Probabilistic Hyperspace Analogue to Language

Probabilistic Hyperspace Analogue to Language (pHAL) normalizes the term co-occurrence matrix by the terms count and marginalizes over all possible values of distance between terms to obtain conditional probability of term co-occurrences. For further information see [9].

2.4 Semantic Similarity between Words

In order to compare two texts, we must first assign a similarity to pairs of words. Most semantic similarity measures use WordNet [12] to determine similarity between words. WordNet is an online lexical reference system developed at Princeton University. Nouns, verbs, adjectives and adverbs are grouped into synonym sets (synsets). The synsets are related by different types of relationships to other synsets higher or lower in the hierarchy. The hierarchical structure of the knowledge base is important for determining the semantic distance between words. [13]

We need to find semantic similarity (also relatedness) for two words denoted as $rel(w_1, w_2)$. Since the knowledge base is hierarchical we think of it as a graph. Then the relatedness between two words can be calculated as the length of the shortest path between them.

$$L = len(w_1, w_2) \tag{2.12}$$

Other methods compute the depth of nodes in the structure. Length of the shortest path from the selected node to the root ($depth(w_i)$). The depth function needs only one node, therefore we use the most specific common subsumer, that subsumes both measured words ($lso(w_i, w_j)$).

$$H = depth(lso(w_1, w_2)) \tag{2.13}$$

Wu and Palmer [26] defined the similarity measure as words position in the lexical hierarchical structure relative to the position of the most specific common subsumer.

$$rel_{W\&P}(w_1, w_2) = \frac{2H}{L + 2H} \tag{2.14}$$

According to [20] relatedness of words is a nonlinear function which combines the shortest path and the depth of the most specific common subsumer. The contribution of path length is a monotonically decreasing function with the value within the range from 0 to 1. Because words at upper layers of hierarchical structure have more general meaning and less semantic similarity than words at lower layers, the contribution of depth part of the function should be an monotonically increasing function.

$$rel_{Li}(w_1, w_2) = e^{-\alpha L} \frac{e^{\beta H} - e^{-\beta H}}{e^{\beta H} + e^{-\beta H}}, \tag{2.15}$$

where $\alpha \in [0, 1], \beta \in (0, 1]$ are parameters scaling the contribution of shortest path length and depth. The optimal values of α and β are dependent on the used knowledge base. Accorting to [20] the optimal parameters for WordNet are $\alpha = 0.2$ and $\beta = 0.45$.

2.5 Sentence Similarity Measures

We can determine similarity between words and large texts, but complete sentence contains more information than just its words. The following measures consider sentence syntax as important information.

2.5.1 Word Overlap Measures

Word overlap measure is a combinatorial similarity measure that computes similarity score based on the number of words shared by two sentences.

Simple Word Overlap and IDF Overlap Measures

Simple word overlap is defined as the number of words that appear in both sentences normalized by the sentence's length. IDF overlap uses inverse document frequency to normalize the result. [8]

Phrasal Overlap Measure

Phrasal overlap measure can be defined as the relation between phrases length and their document frequencies. Traditional word overlap measure treats a sentence as a bag of words and doesn't take into consideration the difference between single words and multi-word phrases. Because n-word overlap is

much rarer than single word overlap, thus m phrasal n-word overlaps are defined as a non-linear function displayed below.

$$overlap_{phrase}(s_1, s_2) = \sum_{i=1}^{n} \sum_{m} i^2, \tag{2.16}$$

where m is the number of i-word phrases that appear in both sentences (s1, s2). The equation can be normalized by the sum of sentences length and applying the hyperbolic tangent function to minimize the effect of outliers. [8] The result of the normalization is shown below.

$$sim_{phrase,overlap}(s_1, s_2) = \tanh \frac{overlap_{phrase}(s_1, s_2)}{|s_1| + |s_2|} \tag{2.17}$$

2.5.2 Linguistic Measures

Linguistic measures use syntactic composition of the sentence or semantic information contained in sentence to determine semantic similarity. [8]

Sentence Semantic Similarity Measure

Li et al. [20] suggest a semantic-vector approach to compute sentence similarity. Sentences are transformed into feature vectors with distinct words from both examined sentences as a feature set T. The value of an entry of the semantic vector is determined by the semantic similarity of the corresponding word from the feature set $w_i \in T$ to a word w_j from the sentence. The word w_j with the highest similarity score $rel(w_i, w_j)$ is selected. The similarity score must exceed preset threshold, otherwise it is set to zero. The assigned value is then weighted by the information weight ($I(w)$) of both words. Finally, the value of an entry of the semantic vector is shown in following equation.

$$s_i = rel(w_i, w_j) * I(w_i) * I(w_j), \tag{2.18}$$

$$I(w) = 1 - \frac{\log(n+1)}{\log(N+1)}, \tag{2.19}$$

where n is the frequency of word w in both sentences, and N is the total number of words in both sentences.

The semantic similarity between two sentences is computed as a cosine similarity between feature vectors of the two sentences.

$$sim_{Li}(s_1, s_2) = \frac{s_1 \times s_2}{\|s_1\| \times \|s_1\|} \tag{2.20}$$

Mihalcea [23] proposes another semantic measure that combines word semantic similarity scores with word specificity scores. For two sentences s_1 and s_2 we find the maximum word similarity score for each word in s_1 with words in the same part of speech class in s_2 and then we repeat the process for sentence s_2. We find the maximum word similarity score for each word in s_2 with words in the same part of speech class in s_1. Then a similarity score is computed according to equation 2.21.

$$sim_{Mi}(s_1, s_2) = \frac{1}{2} \frac{\sum\limits_{w \in \{s_1\}} weightSim(w, s_2)}{\sum\limits_{w \in \{s_1\}} idf(w)} + \frac{\sum\limits_{w \in \{s_2\}} weightSim(w, s_1)}{\sum\limits_{w \in \{s_2\}} idf(w)}, \tag{2.21}$$

$$weightSim(w, s) = maxSim(w, s) * idf(w), \tag{2.22}$$

where $maxSim(w, s_i)$ is the maximum semantic similarity score of w and given sentence and $idf(w)$ is inverse document frequency of w. The semantic similarity scores are computed only between words in the same part of speech class because most measures using lexical databases such as WordNet [12] are unable to compute semantic similarity of cross-part-of-speech words. [8]

Word Order Similarity Measure

Word order similarity measure is defined as normalized difference of word order between two sentences.

$$sim_{wo}(s_1, s_2) = 1 - \frac{||r_1 - r_2||}{||r_1 + r_2||}, \tag{2.23}$$

where r_1 and r_2 are word order vectors of sentences s_1 and s_1. Word order vector r is a feature vector whose feature set T comes from distinct words in a sentence pair. Word order vector r is derived by computing a word similarity score between $w_i \in T$ and all words in sentence. The value of an entry of the word order vector is index of the word w_j from the sentence with the highest similarity score to $w_i \in T$. The maximum word similarity score must exceed preset threshold, otherwise zero value is assigned. [8]

Combined Semantic and Syntactic Measure

Since both semantic and syntactic information describe the sentence, a sentence similarity measure can be defined as a linear combination of semantic

vector similarity and word order similarity. The contribution rate is described by a coefficient. Similarity of two sentences is shown in the following equation.

$$sim_{Li+wo}(s_1, s_2) = \delta sim_{Li}(s_1, s_2) + (1 - \delta) sim_{wo}(s_1, s_2) \tag{2.24}$$

Syntax-based measure for short-text semantic similarity

Syntax-based measure for short-text semantic similarity (SyMSS) computes semantic similarity of two sentences based on semantic information obtained from a lexical database (WordNet [12]) and syntactic information obtained through a deep parsing process. This method overcomes some limitations of WordNet for example the absence of proper nouns. It also outperforms other methods, because it uses complete syntactic information instead of only word order. [24]

2.6 Data Sets

Two publicly-available sentence pair data sets are used to evaluate the performance of the sentence similarity measures. The data sets are Microsoft Research paraphrase corpus (MSRP) [11] and Semantic Textual Similarity (STS) [6] shared task.

2.6.1 MSRP data set

MSRP data set consists of 5801 pairs of sentences automatically constructed from various news sources on the web. These sentences were separated into training data (4076 sentence pairs) and test data (1725 sentence pairs). Accompanying each pair is a judgment reflecting whether multiple human annotators considered the two sentences to be close enough in meaning to be considered semantically equivalent. The structure of the data set files is following:

```
1 Quality #1 ID #2 ID #1 String #2 String
2 0 1227467 1227133 Looking to buy the latest Harry
    Potter? Harry Potter's latest wizard trick?
3 1 690579 690843 I'm never going to forget this day.
     I am never going to forget this throughout my
    life."
```

The first sentence pair is semantically not equivalent (quality = 0). Second sentence is semantically equivalent (quality = 1). Tabulator is used as a separator. Overall, 3900 (67%) of the original 5801 pairs were judged semantically equivalent.

2.6.2 STS data set

Semantic Textual Similarity measures the degree of semantic equivalence. Given two sentences of text, s_1 and s_2, the systems participating in this task should compute how similar s_1 and s_2 are, returning a similarity score, and an optional confidence score.

The data set comprises pairs of sentences drawn from the publicly available data sets used in training:

- MSR-Paraphrase, Microsoft Research Paraphrase Corpus
  750 pairs of sentences
- MSR-Video, Microsoft Research Video Description Corpus
  750 pairs of sentences
- SMTeuroparl: WMT2008 develoment dataset (Europarl section)
  734 pairs of sentences
- SMTnews: news conversation sentence pairs from WMT
  399 pairs of sentences
- OnWN: pairs of sentences where the first comes from OntoNotes [15] and the second from a WordNet definition.
  750 pairs of sentences
- ALL: concatenation of the five sources above (different sentence pairs, SMTeuroparl has only 459 pairs of sentences).
  3108 pairs of sentences

The output of participant systems is compared to the gold standard manual scores, which range from 5 (semantic equivalence) to 0 (no relation) for each pair of sentences, with the following interpretation:

5: The two sentences are completely equivalent, as they mean the same thing.

4: The two sentences are mostly equivalent, but some unimportant details differ.

3: The two sentences are roughly equivalent, but some important information differs/missing.

2: The two sentences are not equivalent, but share some details.

1: The two sentences are not equivalent, but are on the same topic.

0: The two sentences are on different topics.

2.7 Evaluation Criteria

Since the STS data set official score is based on Pearson's correlation, we chose it as evaluation criteria as well. Correlation isn't the best choice for MSRP data set, because the data set's semantic equivalence rate is only true or false, thus we selected accuracy as the main evaluation criteria.

	Similar	**Not similar**
classified similar	true positives (TP)	false positives (FP)
classified not similar	false negative (FN)	true negative (TN)

Tab. 2.1: Contingency table.

2.7.1 Accuracy

Accuracy is a proportion of all correctly predicted sentences compared to all sentences. However, this metric is not useful if we need to reject as many dissimilar sentences as possible or accept all similar sentences even if it means accepting some dissimilar sentences. That is why we selected also rejection rate and acceptance rate as evaluation criteria.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.25}$$

2.7.2 Rejection Rate

Rejection rate is a proportion of correctly predicted dissimilar sentences compared to all dissimilar sentences. High acceptance rate means that we recognized almost all dissimilar sentences, but it doesn't tell us how many similar sentences we recognized. That is why we also investigate acceptance rate.

$$Rejection\,rate = \frac{TN}{TN + FP} \tag{2.26}$$

2.7.3 Acceptance Rate

Acceptance rate is a proportion of correctly predicted similar sentences compared to all similar sentences. High acceptance rate means that we recognized almost all similar sentences, but it doesn't tell us how many dissimilar sentences we recognized. That is why we investigate all three metrics (accuracy, rejection rate and acceptance rate).

$$Acceptance\,rate = \frac{TP}{TP + FN} \tag{2.27}$$

2.7.4 Pearson's Correlation

Correlation is one of the most common and most useful statistics. Correlation between two variables reflects the degree to which the variables are related. The most common measure of correlation is the Pearson Product Moment Correlation (Pearson's correlation) which is described by the following equation.

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} = \frac{\sum_{i=1}^{n}(x_i y_i) - \frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n}}{\sqrt{(\sum_{i=1}^{n} x_i^2 - \frac{\sum_{i=1}^{n}(x_i)^2}{n})(\sum_{i=1}^{n} y_i^2 - \frac{\sum_{i=1}^{n}(y_i)^2}{n})}} \tag{2.28}$$

$$= \frac{n\sum_{i=1}^{n}(x_i y_i) - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{n\sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n}(x_i)^2}\sqrt{n\sum_{i=1}^{n} y_i^2 - \sum_{i=1}^{n}(y_i)^2}} \tag{2.29}$$

Pearson's correlation reflects the degree of linear relationship between two variables. It ranges from +1 to -1. A correlation of +1 means that there is a perfect positive linear relationship between variables. A correlation of -1 means that there is a perfect negative linear relationship between variables. A correlation of 0 means there is no linear relationship between the two variables. As it approaches zero, the relationship gets less significant (closer to uncorrelated). The closer the coefficient is to either -1 or 1, the stronger the correlation between the variables.

3 Implemented State-of-the-art Similarity Methods

We selected promising methods from the Semantic similarities between Words (section 2.4) and Sentence Similarity Measures (section 2.5) and implemented them. Short description of these methods is in section 3.1 and section 3.2.

3.1 Token Similarities

Token similarity defines semantic similarity (relatedness) between two tokens. As described in section 2.4 WordNet is a lexical reference system developed at Princeton University. We use WordNet to determine the semantic distance between tokens. Since the knowledge base is hierarchical we can think of it as a graph. Than the relatedness between two tokens can be calculated according to the methods described in section 2.4.

3.1.1 Basic Token Similarity

Basic Token Similarity compares two tokens by their part of speech tag and their content and returns one if they are equal or zero when they differ. It does not use WordNet and we use it only as a base line for token similarities.

3.1.2 WordNet Token Similarity

WordNet Token Similarity is a representation of similarity measure described in equation 2.15.

3.1.3 WordNet Wu Palmer Token Similarity

WordNet Wu Palmer Token Similarity is a representation of similarity measure described in equation 2.14.

3.2 Sentence Similarities

In this section we will describe implemented methods from sentence similarity measures. We selected promising well implementable methods from section 2.5. As a representant of word overlap measures we chose *Phrasal Overlap*

Measure. From section 2.5.2 all linguistic measures except SyMSS were selected. SyMSS is too extensive for implementation and performs deep parsing process that would take a significant computational time.

3.2.1 Phrasal Overlap Measure

Phrasal Overlap Measure is calculated according to equation 2.17. It was selected as a representant of word overlap measures.

3.2.2 Sentence Semantic Similarity

The implementation of *Sentence Semantic Similarity* is based on equation 2.20. The threshold was set to 0.2 as described in [20].

3.2.3 Mihalcea Semantic Similarity

Mihalcea Semantic Similarity is computed according to equation 2.21.

3.2.4 Word Order Similarity

Word Order Similarity is determined by equation 2.23. The threshold was set to 0.4 according to [20].

3.2.5 Combined Semantic Syntactic Measure

Combined Semantic Syntactic Measure is calculated according to equation 2.24. We set the δ parameter to 0.85 because according to [8] it is the best value for δ.

4 Enhanced Similarity Methods

In this section we will describe our proposed similarity methods inspired by methods mentioned in chapter 3.

4.1 Enhanced WordNet Token Similarity

Token similarity defines semantic similarity (relatedness) between two tokens. The relatedness between two tokens can be calculated according to methods described in the section 2.4. We were inspired by these methods and tried to propose an enhanced token similarity measure.

We empirically developed an *Enhanced WordNet Token Similarity* as a nonlinear function of depth of the most specific common subsumer and shortest path length between the two tokens. Instead of the previous denominator we normalized the result by the product of depth of the most specific common subsumer and shortest path length between the two tokens and applied the hyperbolic tangent function.

$$rel_E(w_1, w_2) = \tanh \frac{2H}{L * H + 1} \tag{4.1}$$

4.2 Enhanced Sentence Similarities

We adjusted or combined sentence similarities from section 3.2 to achieve better results. In this section we will describe the modifications.

4.2.1 Enhanced Phrasal Overlap Measure

Phrasal Overlap Measure (section 2.5.1) doesn't use WordNet-based token similarity. We think that *Enhanced Phrasal Overlap Measure* could achieve better results with more accurate similarity score from WordNet-based token similarity. We modified it and now it uses WordNet token similarity. *Enhanced Phrasal Overlap Measure* is described by equation 4.2.

$$sim_{Epo}(s_1, s_2) = \tanh\left(\frac{sum_{po}(s_1, s_2)}{|s_1|} + \frac{sum_{po}(s_2, s_1)}{|s_2|}\right), \tag{4.2}$$

where $sum_{po}(s_1, s_2)$ is computed by the following pseudocode.

```
1 INIT result to zero
2 INIT sum to zero
```

```
3  FOR i = 0 to sentence1 length
4      INIT commonWords to zero
5      FOR each token in sentence 2 as token2
6          IF (i + commonWords) < sentence1 length
              THEN
7              SET token1 to sentence1 [i +
                  commonWords]
8              IF POS of token1 equals POS of token2
                  THEN
9                  CALL similarity with token1 and
                      token2
10                 RETURNING similarity
11             ELSE
12                 IF token1 equals token2 THEN
13                     SET similarity to one
14                 ELSE
15                     CONTINUE
16                 END IF
17             END IF
18             IF similarity > threshold THEN
19                 increment commonWords
20                 ADD similarity into sum
21             ELSE
22                 SET commonWords to zero
23                 SET sum to zero
24             END IF
25         ELSE
26             SET commonWords to zero
27             SET sum to zero
28         END IF
29         ADD sum square into result
30     END FOR
31 END FOR
32 RETURN result
```

Since WordNet-based token similarities compute similarity only between tokens in the same part of speech class, we adjusted this method. If the compared tokens are in different part of speech class, then the similarity score is computed according to *Basic Token Similarity* in section 3.1.1. When the compared tokens are in different part of speech classes and they are not similar, then we move on to next token pair. We added a threshold that must

be exceeded, so that we can consider the two tokens similar and add their similarity score to the sum. The best average of results is achieved with the threshold equal to 0.8.

4.2.2 Enhanced Sentence Semantic Similarity

We removed threshold from *Sentence Semantic Similarity*, because we are looking for the maximum similarity and even if this similarity is close to zero, we consider it significant. We also removed token information weight ($I(w)$) because in a document composed of two sentences it is mostly constant.

As in *Sentence Semantic Similarity* (section 2.5.2) the sentences are transformed into feature vectors with distinct words from both examined sentences as a feature set T. The value of an entry of the semantic vector is determined by the semantic similarity of the corresponding word from the feature set $w_i \in T$ to a word w_j from the sentence. The highest similarity score $rel(w_i, w_j)$ is assigned to the feature vector.

The value of an entry of the semantic vector is shown in following equation.

$$s_i = rel(w_i, w_j) \tag{4.3}$$

The semantic similarity between sentences is computed as a cosine similarity between feature vectors of the two sentences.

$$sim_{ELi}(s_1, s_2) = \frac{s_1 \times s_2}{\|s_1\| \times \|s_1\|} \tag{4.4}$$

4.2.3 Enhanced Mihalcea Similarity

We removed $idf(w)$ from *Mihalcea Semantic Similarity*, because in a document composed of two sentences it is mostly constant. The similarity score is normalized by sentences length and tanh function. Similarity score is computed according to equation 4.5.

$$sim_{EMi}(s_1, s_2) = \tanh\left(\frac{\sum\limits_{w \in \{s_1\}} maxSim(w, s_2)}{|s_1|} + \frac{\sum\limits_{w \in \{s_2\}} maxSim(w, s_1)}{|s_2|}\right), \tag{4.5}$$

where $maxSim(w, s_i)$ is the maximum semantic similarity score of w and given sentence.

As in *Mihalcea Semantic Similarity* (section 2.5.2) the semantic similarity scores are computed only between tokens in the same part of speech class

because WordNet is unable to compute semantic similarity between tokens in different part of speech classes. [8]

4.2.4 Enhanced Combined Semantic Syntactic Measure

This measure consists of adapted *Sentence Semantic Similarity* and *Word Order Similarity* (section 2.5.2).

We have removed threshold from *Sentence Semantic Similarity*, because we are looking for the maximum similarity and even if this similarity is close to zero, we consider it significant.

As in *Sentence Semantic Similarity* the sentences are transformed into feature vectors with distinct words from both examined sentences as a feature set T. The value of an entry of the semantic vector is determined by the semantic similarity of the corresponding word from the feature set $w_i \in T$ to a word w_j from the sentence. The word w_j with the highest similarity score $rel(w_i, w_j)$ is selected. The assigned value is then weighted by the information weight ($I(w)$) of both words. Finally, the value of an entry of the semantic vector is shown in following equation.

$$s_i = rel(w_i, w_j) * I(w_i) * I(w_j), \tag{4.6}$$

$$I(w) = (1 - \frac{\log(n+1)}{\log(N+1)}) * POSweight(w), \tag{4.7}$$

where n is the frequency of word w in both sentences, N is the total number of words in both sentences and $POSweight(w)$ is part of speech weight, which we assigned the value 1 for adjective and verb, 0.7 for adverb and noun, 0.25 otherwise.

The semantic similarity between sentences is computed as a cosine similarity between feature vectors of the two sentences.

$$sim_{ECLi}(s_1, s_2) = \frac{s_1 \times s_2}{\|s_1\| \times \|s_1\|} \tag{4.8}$$

We kept the *Word Order Similarity* unchanged, we only selected different similarity threshold.

The best average results are achieved with a threshold equal to 0.8. The similarity measure is described by equation 4.9. The contribution rate is described by δ coefficient. The best average results are achieved with coefficient δ equal to 0.55.

$$sim_{ECLi+wo}(s_1, s_2) = \delta sim_{ECLi}(s_1, s_2) + (1 - \delta) sim_{wo}(s_1, s_2) \tag{4.9}$$

4.2.5 Combined Semantic Phrasal Overlap Measure

We have selected promising linguistic measure and word overlap measure and combined them in one similarity measure. *Enhanced Sentence Semantic Similarity* and *Enhanced Phrasal Overlap Measure* were chosen for this purpose because they could complement each other. Their combination is described by following equation. The best average results are achieved with coefficient δ equal to 0.85.

$$sim_{ELi+Epo}(s_1, s_2) = \delta sim_{ELi}(s_1, s_2) + (1 - \delta) sim_{Epo}(s_1, s_2) \tag{4.10}$$

4.2.6 Combined Mihalcea Phrasal Overlap Measure

We continued with experimenting and combined *Enhanced Mihalcea Similarity* with *Enhanced Phrasal Overlap Measure*. Equation 4.11 defines their combination. The best average results are achieved with coefficient δ equal to 0.4.

$$sim_{EMi+Epo}(s_1, s_2) = \delta sim_{EMi}(s_1, s_2) + (1 - \delta) sim_{Epo}(s_1, s_2) \tag{4.11}$$

5 Implementation

In this section we will describe implementation and architecture of our project. We used Java as programming language because of its modularity, robustness, scalability and high availability of libraries and tools for development.

5.1 Architecture

In order to evaluate similarity measures on data sets we must first load the data sets into object representation that is displayed in figure 5.1. Data set is represented by a list of *Documents* (in our case sentence pairs). Sentence consists of tokens.

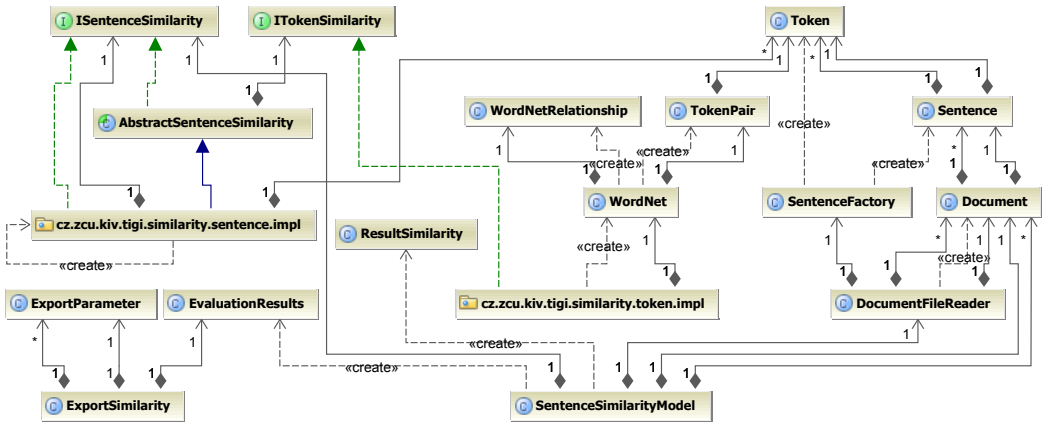

Fig. 5.1: Simplified class diagram

To load and test data set, we must create *SentenceSimilarityModel* with *DocumentFileReader* and *SentenceFactory*. *DocumentFileReader* reads lines from data set files. Structure of the MSRP data set is described in section 2.6.1. STS data set consists of two files (input file and gold standard file). Input file has a sentence pair on each line separated by tabulator. Each line in the gold standard file contains a gold standard manual score for corresponding sentence pair in the input file. *DocumentFileReader* reads lines from data set files and creates their object representation. *SentenceFactory* generates a sentence composed of tokens from given text. We must assign a *SentenceSimilarity* (an implementation of *ISentenceSimilarity* interface) to the model for evaluation. The model is initialized by following command.

```
1 SentenceSimilarityModel model = new
     SentenceSimilarityModel(new DocumentFileReader(
     new SentenceFactory()));
2 model.setSentenceSimilarity(sentenceSimilarity);
```

The following instructions load MSRP data set files and compute accuracy, rejection rate and acceptance rate of examined sentence similarity measure.

```
1 model.train(trainFile);
2 testResults = model.test(testFile);
```

The next command loads STS data set files and computes correlation between gold standard and result of the examined sentence similarity measure.

```
1 correlation = model.getScores(inputFilePath,
     goldStandardFilePath);
```

5.2 Preprocessing

For each data set we perform a tokenization and part-of-speech tagging on a sentence using part of Stanford Parser [10] *PTBTokenizer* and *MaxentTagger* [25]. Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech tags to each word (token), such as noun, verb, adjective, etc.

To access WordNet dictionary we decided to use an application programming interface (API). We found three solutions.

JAWS Java API for WordNet Searching [1] doesn't have the option to load WordNet dictionary into memory and without it the search for relationship between two tokens is too slow.

JWI Java WordNet Interface [3] has a thorough User's Guide and has the ability to load a dictionary fully into memory, however we found its configuration too difficult.

JWNL Java WordNet Library [4] uses a properties xml file, where the used WordNet dictionary is described and we can easily choose between *FileBackedDictionary* and *MapBackedDictionary*. *MapBackedDictionary* takes longer to load (about 40 seconds on Intel Core i5-430M, 4GB

RAM, JDK 1.6.0_20, windows 7), but it provides substantial performance improvement.

We chose Java WordNet Library because of its easy configuration through properties file and its speed. We use *MapBackedDictionary* that requires an map representation of WordNet dictionary. We can build it with the following command.

```
1 java -cp jwnl-1.4_rc3.jar;commons-logging.jar net.
    didion.jwnl.utilities.DictionaryToMap
    targetFolder properties.xml
```

The properties file describes *FileBackedDictionary* of WordNet. We use WordNet version 2.1 and the properties file resembles following example.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <jwnl_properties language="en">
3      <version publisher="Princeton" number="2.1"
         language="en"/>
4      <dictionary class="net.didion.jwnl.dictionary.
         FileBackedDictionary">
5  ...
6          <param name="dictionary_element_factory"
             value="net.didion.jwnl.princeton.data.
             PrincetonWN17FileDictionaryElementFactory
             "/>
7          <param name="file_manager" value="net.
             didion.jwnl.dictionary.file_manager.
             FileManagerImpl">
8            <param name="file_type" value="net.
               didion.jwnl.princeton.file.
               PrincetonRandomAccessDictionaryFile"
               />
9            <param name="dictionary_path" value="
               resources/wordnet/dict_2_1/"/>
10         </param>
11     </dictionary>
12     <resource class="PrincetonResource"/>
13 </jwnl_properties>
```

However, we use *MapBackedDictionary* representation and the properties file resembles following truncated example.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <jwnl_properties language="en">
3      <version publisher="Princeton" number="2.1"
          language="en"/>
4      <dictionary class="net.didion.jwnl.dictionary.
          MapBackedDictionary">
5  ...
6          <param name="file_type" value="net.didion.
             jwnl.princeton.file.
             PrincetonObjectDictionaryFile"/>
7          <param name="dictionary_path" value="
             resources/wordnet/dictMap_2_1/"/>
8      </dictionary>
9      <resource class="PrincetonResource"/>
10 </jwnl_properties>
```

We use WordNet to find relationships between two tokens. The results of the search are the length of the shortest path between the two tokens and depth of the most specific common subsumer of the tokens. Both these values are wrapped in *WordNetRelationship*. Because the search in WordNet takes a significant time we developed a cache for *WordNetRelationship* between two tokens (*TokenPair*), which has sped up the process.

5.3 Project Execution

The evaluation of the proposed similarity methods is computed in *SentenceSimilarityModel* according to section 2.7. To export the evaluation results to excel file we use Java Excel API [2] and for export into graph we use JFreeChart library [5]. The evaluation of proposed similarity methods on MSRP data set can be run by the following code.

```
1  SentenceSimilarityModel model = new
      SentenceSimilarityModel(new DocumentFileReader(
      new SentenceFactory()));
2  List<ExportSimilarity> results = new ArrayList<
      ExportSimilarity>();
3  for (ITokenSimilarity tokenSimilarity :
      getTokenSimilarities()) {
4      for (ISentenceSimilarity sentenceSimilarity :
          getSentenceSimilarities(tokenSimilarity)) {
```

```
5           model.setSentenceSimilarity(
               sentenceSimilarity);
6           model.train(trainFile);
7           EvaluationResults testResults = model.test(
               testFile);
8
9           ExportSimilarity exportSimilarity = new
               ExportSimilarity(testResults,
               sentenceSimilarity.getClass(),
               tokenSimilarity.getClass();
10          results.add(exportSimilarity);
11      }
12  }
13  new Excel("results/accuracy.xls", results);
14  new Graph().saveGraph("results/Accuracy.svg",
       results, "Title", "Accuracy", 1024, 600, false);
```

The evaluation of proposed similarity methods on STS data set can be run by the following code.

```
1  SentenceSimilarityModel model = new
       SentenceSimilarityModel(new DocumentFileReader(
       new SentenceFactory()));
2  List<ExportSimilarity> results = new ArrayList<
       ExportSimilarity>();
3
4  for (ITokenSimilarity tokenSimilarity :
       getTokenSimilarities()) {
5       for (ISentenceSimilarity sentenceSimilarity :
           getSentenceSimilarities(tokenSimilarity)) {
6           model.setSentenceSimilarity(
               sentenceSimilarity);
7           double correlation = model.getScores(
               inputFilePath, gsFilePath, multiplier,
               addConst);
8
9           ExportSimilarity exportSimilarity = new
               ExportSimilarity(correlation,
               sentenceSimilarity.getClass(),
               tokenSimilarity.getClass());
```

```
10          exportSimilarity.addParametr(new
               ExportParameter(0, inputFilePath));
11          results.add(exportSimilarity);
12      }
13  }
14  new Excel("results/correlation.xls", results);
15  new Graph().saveGraph("results/correlation.svg",
       results, "Title", "Correlation", 1024, 600, false
       );
```

Both previous examples create excel file and graph with the evaluation results of proposed similarity methods.

The project is build on JUnit[18] tests. The evaluation can be run through tests in class *SimilarityTest*. The correct project execution is ensured by tests in class *FunctionalityTest*. Each test checks functionality of part of the project.

Whole project can be build and tests executed with the following Maven command.

```
1 mvn test
```

This command creates directory results with exported evaluation results (graphs and excel files).

6 Evaluation

Our task was to evaluate the proposed methods on two publicly-available data sets. The data sets are Microsoft Research paraphrase corpus(MSRP) and Semantic Textual Similarity (STS) shared task. We described them in section 2.6.

We use different evaluation criteria on each data set. MSRP data set assigns to a sentence pair only two values (semantically equivalent or semantically not equivalent), thus we chose accuracy (section 2.7.1) as the main evaluation criteria. Rejection rate (section 2.7.2) and acceptance rate (section 2.7.3) are additional metrics used to evaluate MSRP data set. For STS data set the evaluation criteria was given. The STS shared task has Pearson's correlation (section 2.7.4) as it's official score, so we chose correlation as well.

The STS data set requires wider range of values than our system provides, that is why we transform our results to fit in the requested range. Our result ranges from 0 to 1 while STS data set ranges from 0 to 5, therefore we multiply our outcome by 5.

The MSRP data set allows only two values (semantically equivalent and semantically not equivalent), therefore we introduce a threshold, which separates our results into two categories according to MSRP requirements. We identify the threshold on the train data and study the results on test data from the data set.

Some of the similarity methods require parameters to be determined in advance. Thresholds and factor δ for weighting the significance between two combined similarity measures were experimentally found using MSRP data set.

We present the performance of the evaluated methods in a simple stripe chart. The newly proposed methods are distinguished in graphs by blue color.

6.1 Data Analysis

Since sentence similarity measures strongly depend on the examined data, we present analysis of all used data sets in table 6.1.

The effectiveness of linguistic measures depends on a heuristic to compute semantic similarity between words as well as the comprehensiveness of the lexical resource. The comprehensiveness is determined by the proportion of words in data sets that are covered by knowledge base of used lexical database. As indicated on table 6.1 the coverage in WordNet decreases as the size of data set and vocabulary space increases. Thus, the effectiveness of

Description	Data set							
	MSRP		STS					
	train	**test**	**ALL**	**MSRpar**	**MSRvid**	**SMTeur**	**OnWN**	**SMTnews**
Number of sentence pairs	4076	1725	3108	750	750	459	750	399
Number of unique words	16836	10456	9171	6058	1203	527	2791	823
Number of unique words covered by WordNet	7647	5053	4997	2966	641	305	2036	456
Percentage of unique words covered by WordNet	45.42%	48.33%	54.49%	48.96%	53.28%	57.87%	72.95%	55.41%
Average sentence length (in characters)	98.36	98.00	53.69	92.03	25.96	55.05	37.95	61.77
Average difference in length of a sentence pair (in characters)	16.51	16.55	11.57	17.90	5.76	6.42	16.04	8.11
Average sentence length (in tokens)	21.90	21.72	12.48	20.59	7.64	12.30	8.79	13.48
Average difference in length of a sentence pair (in tokens)	3.46	3.51	2.67	3.69	1.35	1.78	3.87	2.01

Tab. 6.1: Data sets analysis.

linguistic measures is likely to be affected because calculation of semantic similarity between words will inevitably produce many "misses". One solution is to utilize other knowledge resources to derive semantic similarity between words. [8]

6.2 MSRP data set

As mentioned in section 2.7, we have chosen accuracy as the main evaluation criteria for MSRP data set. Figure 6.1 shows the accuracy of the proposed semantic similarity methods. We also examined rejection rate and acceptance rate of these methods (figure 6.2 and 6.3).

6.2.1 Accuracy

Accuracy of proposed similarity methods is shown in figure 6.1. The *Enhanced Combined Semantic Syntactic Measure* achieves the best accuracy closely followed by *Combined Semantic Syntactic Measure*. Word overlap measures do not perform so well, because of the structure of the data set and because it doesn't use the semantic information contained in the sentence, only the

structure of the sentence. There aren't so many phrasal word overlaps in sentence pairs.

Mihalcea Semantic Similarity compares only tokens in the same part of speech class, thus it depends on the used lexical knowledge base. WordNet database is quite limited because it covers only adverbs, adjectives, nouns and verbs (precise percentage of unique words coverage is shown in table 6.1). Previous reasons explain why sentence similarity measures based on *Mihalcea Semantic Similarity* don't perform so well as others. The only exception is the combination of measures based on *Mihalcea Semantic Similarity* with *Basic Token Similarity*, because it compares tokens based on their part of speech tag and their content instead of using WordNet database.

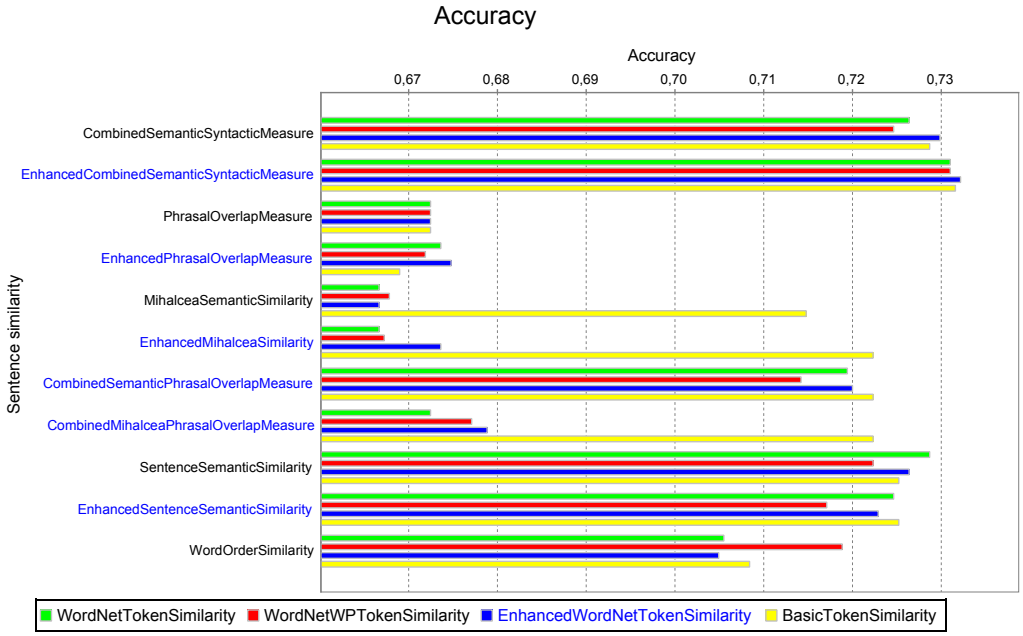

Fig. 6.1: Accuracy of examined similarity methods on MSRP data set.

6.2.2 Rejection Rate

Rejection rate of examined similarity methods is shown in figure 6.2. Although sentence similarity measures based on *Mihalcea Semantic Similarity* and *Enhanced Phrasal Overlap Measure* (with *Basic Token Similarity*) don't achieve very high accuracy, they achieve significantly better rejection rate. Thus these methods are better at dissimilar sentences recognition than the other methods. The rest of the similarity methods produce equivalent results.

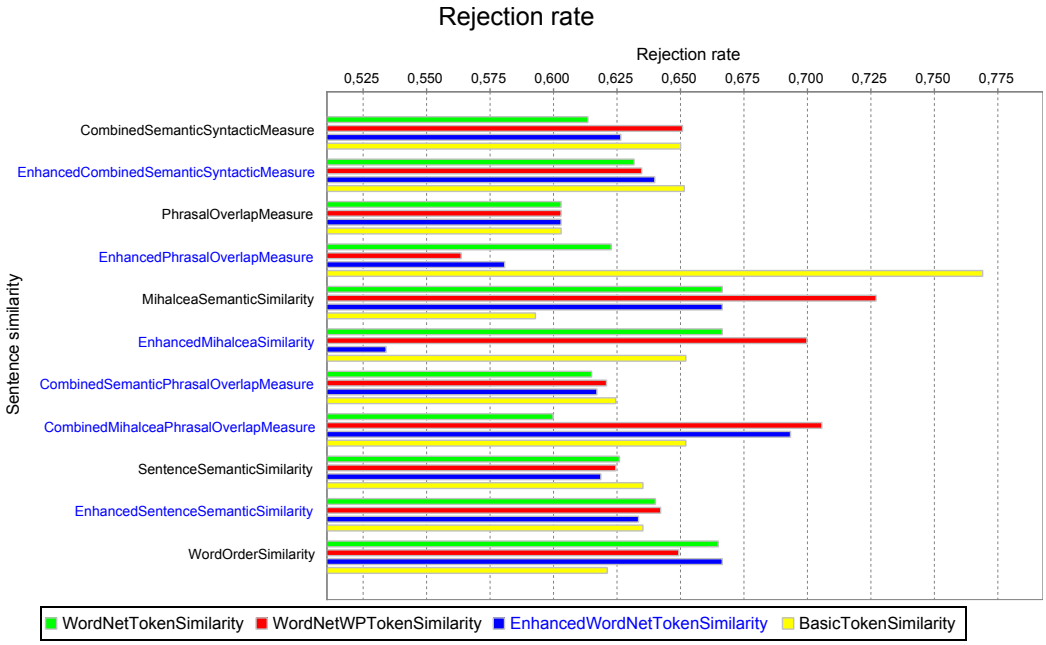

Fig. 6.2: Rejection rate of examined similarity methods on MSRP data set.

6.2.3 Acceptance Rate

Acceptance rate (figure 6.3) and accuracy of examined similarity methods have similar behavior with only slight differences. Word overlap measures do not perform so well, because of the structure of the data set and because they don't use the semantic information contained in the sentence, only the structure of the sentence. There aren't so many phrasal word overlaps in sentence pairs. *Mihalcea Semantic Similarity* is limited by tokens in the same part of speech class, thus it depends on the used lexical knowledge base and doesn't perform so well as other methods. The only exception is the combination of measures based on *Mihalcea Semantic Similarity* with *Basic Token Similarity*, because it compares tokens based on their part of speech tag and their content instead of using WordNet database.

6.3 STS data set

As was mentioned in section 2.7, correlation is the official score for STS data set. Figure 6.4 shows the correlation of the proposed semantic similarity methods on the STS.ALL data set.

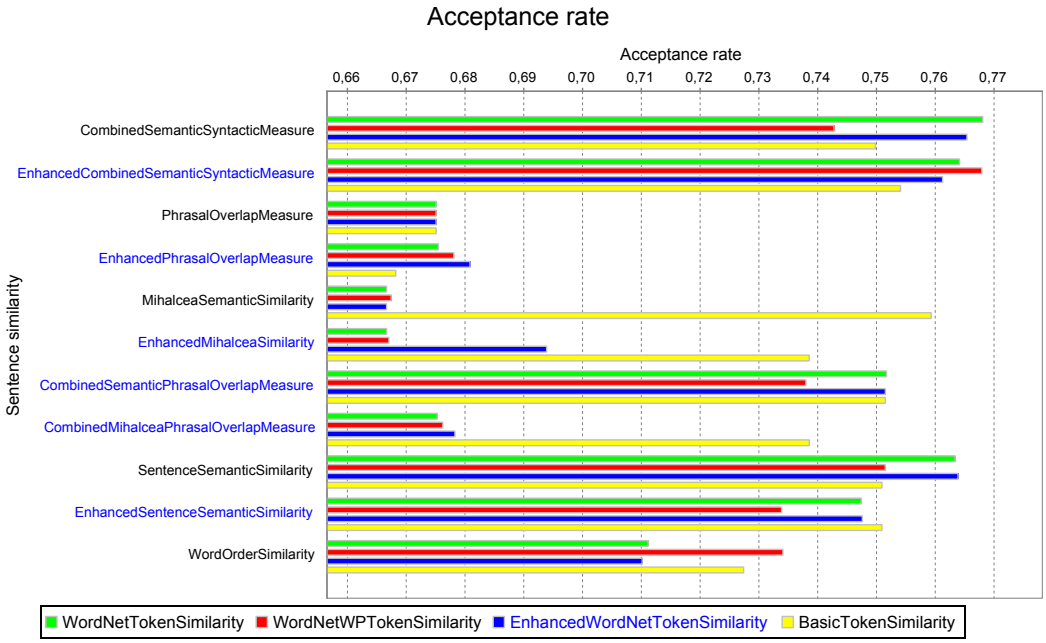

Fig. 6.3: Acceptance rate of examined similarity methods on MSRP data set.

6.3.1 STS ALL data set

The best similarity measure on the STS.ALL data set is certainly *Combined Mihalcea Phrasal Overlap Measure* with *Enhanced WordNet Token Similarity* (correlation 0.4594). The second best similarity measure is *Enhanced Mihalcea Similarity* with *Enhanced WordNet Token Similarity* (correlation 0.4251). Similarities based on *Mihalcea Semantic Similarity* and *Enhanced Phrasal Overlap Measure* achieve good results. These measures are limited by the proportion of words in data sets that are covered by knowledge base of used lexical database (WordNet in our case). As mentioned in section 6.1 STS.ALL has only 54.49% of unique words covered by WordNet.

6.3.2 STS MSRpar data set

On the STS.MSRpar data set the behavior of correlation of similarity measures is similar to the behavior of accuracy of these methods because the STS.MSRpar data set is a part of MSRP dataset. The best results (figure 6.5) were achieved by similarity measures based on *Sentence Semantic Similarity*.

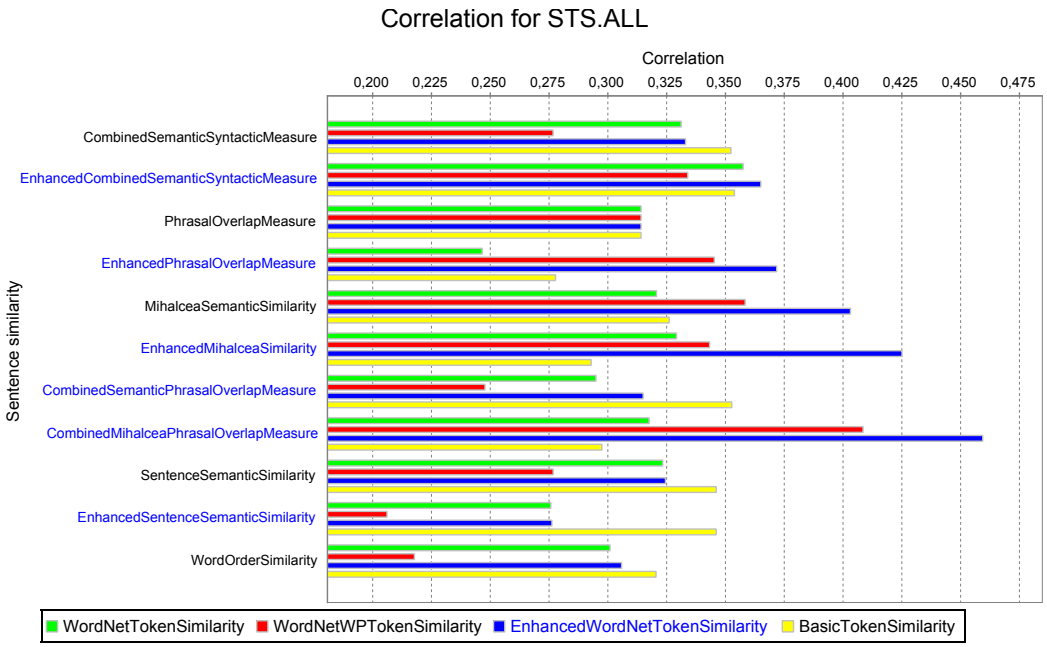

Fig. 6.4: Correlation between STS.ALL and results of examined similarity methods.

6.3.3 STS MSRvid data set

The best results on STS.MSRvid data set (figure 6.6) were achieved by *Mihalcea Semantic Similarity* and *Enhanced Mihalcea Similarity*. It seems that similarity measures based on *Mihalcea Semantic Similarity* are affected by difference in length of sentence pairs because they perform quite well on data sets with low average difference in length of sentence pairs. Based on data analysis from section 6.1, the lowest average difference in length of sentence pairs are in STS.MSRvid data set (5.76 characters), STS.SMTeur data set (6.42 characters), STS.SMTnews data set (8.11 characters) and STS.ALL data set (11.57 characters). Other similarity methods produce essentially equivalent results.

6.3.4 STS SMTeur data set

On the STS.SMTeur data set (figure 6.7) the behavior of correlation of similarity measures is basically equivalent except for *Mihalcea Semantic Similarity* and *Enhanced Mihalcea Similarity* with WordNet based token similarity measures that don't perform so well. This result can be explained by the high sentences length (55.05 characters). The good performance of other methods is caused by the high coverage of words by WordNet (57.87%).

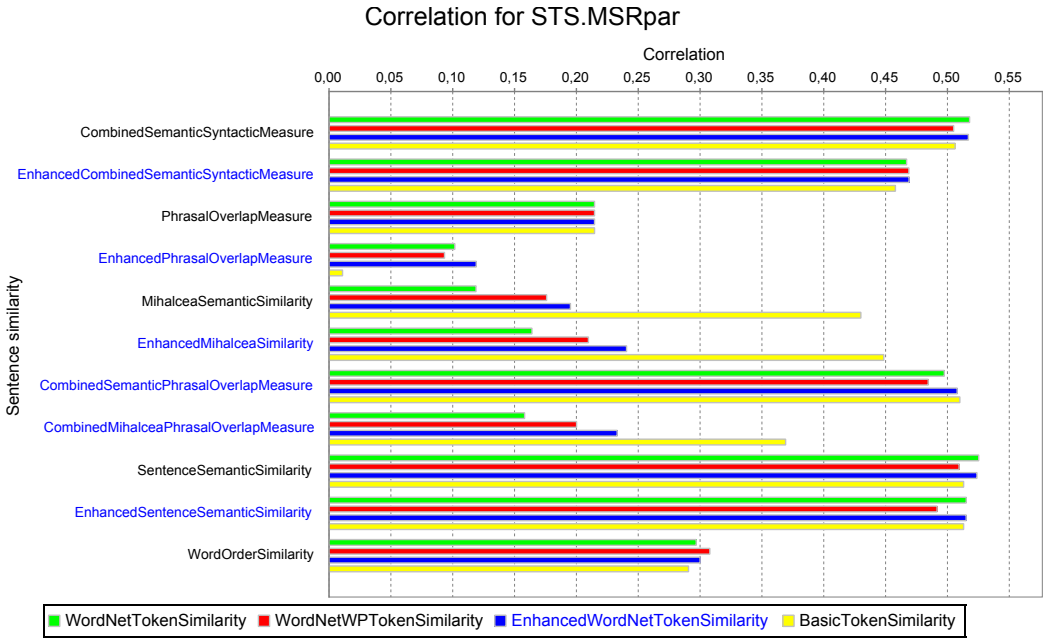

Fig. 6.5: Correlation between STS.MSRpar and the results of the examined similarity methods.

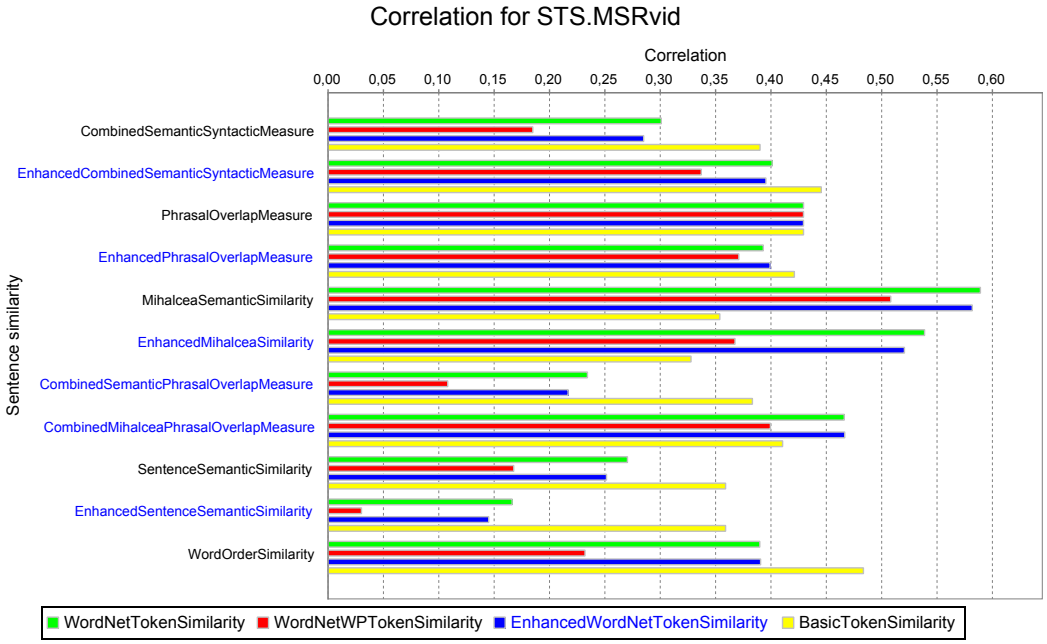

Fig. 6.6: Correlation between STS.MSRvid and the results of the examined similarity methods.

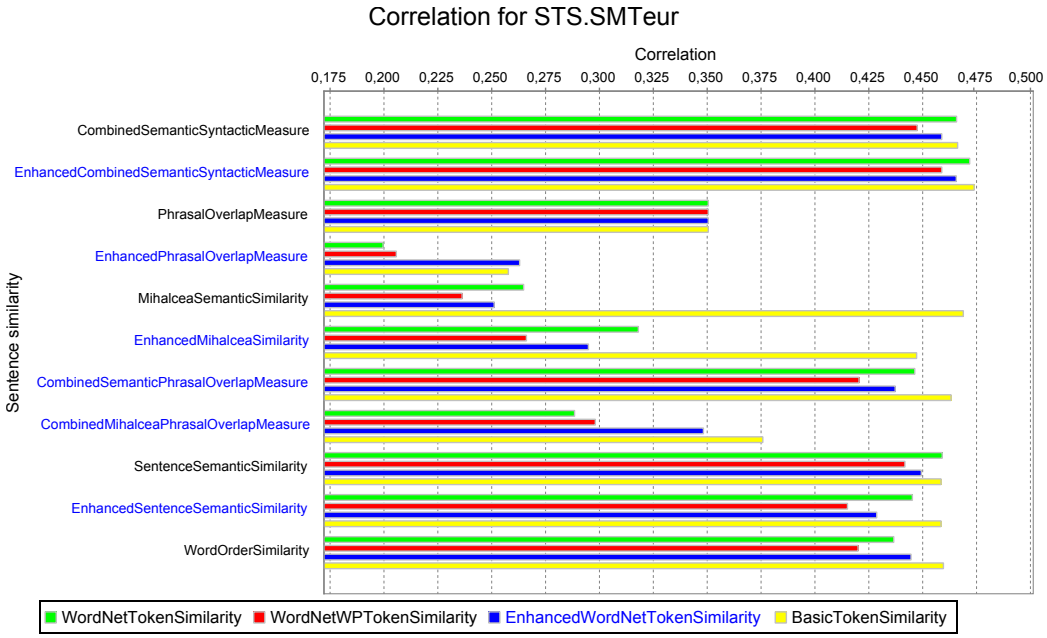

Fig. 6.7: Correlation between STS.SMTeur and the results of the examined similarity methods.

6.3.5 STS OnWN data set

On the STS.OnWN data set (figure 6.8) the behavior of correlation of similarity measures is basically equivalent except for *Mihalcea Semantic Similarity* and *Enhanced Mihalcea Similarity* with WordNet based token similarity measures that perform quite poorly. We explain this result by the high average difference in length of a sentence pair. Other methods perform as well as on STS.SMTeur because coverage of words by WordNet is very good (according to table 6.1 it is 72.95%). We explain this behavior by the short average length of sentences (37.95 characters) and the data composition. First sentence comes from OntoNotes [15](The OntoNotes project is creating a corpus of large scale, accurate, and integrated annotation of multiple levels of the shallow semantic structure in text.) and the second from WordNet definition. The data composition ensures that words will occur repeatedly. Results with *Basic Token Similarity* as token similarity achieve better correlation because it compares tokens based on their part of speech tag and their content instead of using WordNet database.

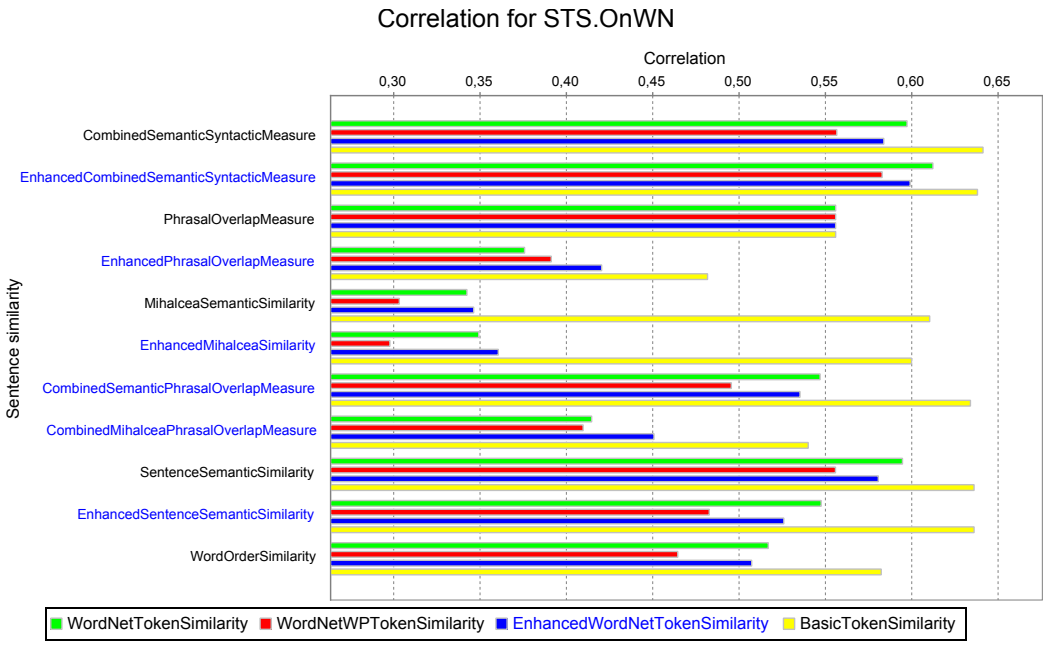

Fig. 6.8: Correlation between STS.OnWN and the results of the examined similarity methods.

6.3.6 STS SMTnews data set

On the STS.SMTnews data set (figure 6.9) the behavior of correlation of similarity measures is basically equivalent except for *Mihalcea Semantic Similarity* and *Enhanced Mihalcea Similarity* with *WordNet Token Similarity* that perform quite poorly. We explain this result by the high sentences length and high average difference in length of a sentence pair. *Combined Mihalcea Phrasal Overlap Measure* with *WordNet Wu Palmer Token Similarity* or *Enhanced WordNet Token Similarity* perform very well on this data set.

6.3.7 STS Task's results

The STS task published results on their site [7]. They evaluated 89 systems including their baseline. If we had participated in this task then we would have placed on the 70th position out of 90 participants. On table 6.2 are shown the results of the STS task in comparison to our best result (*Combined Mihalcea Phrasal Overlap Measure* with *Enhanced WordNet Token Similarity*).

The rank was awarded according to correlation for STS.ALL data set. Our best result is 0.4594 and it is better than the baseline and it would be

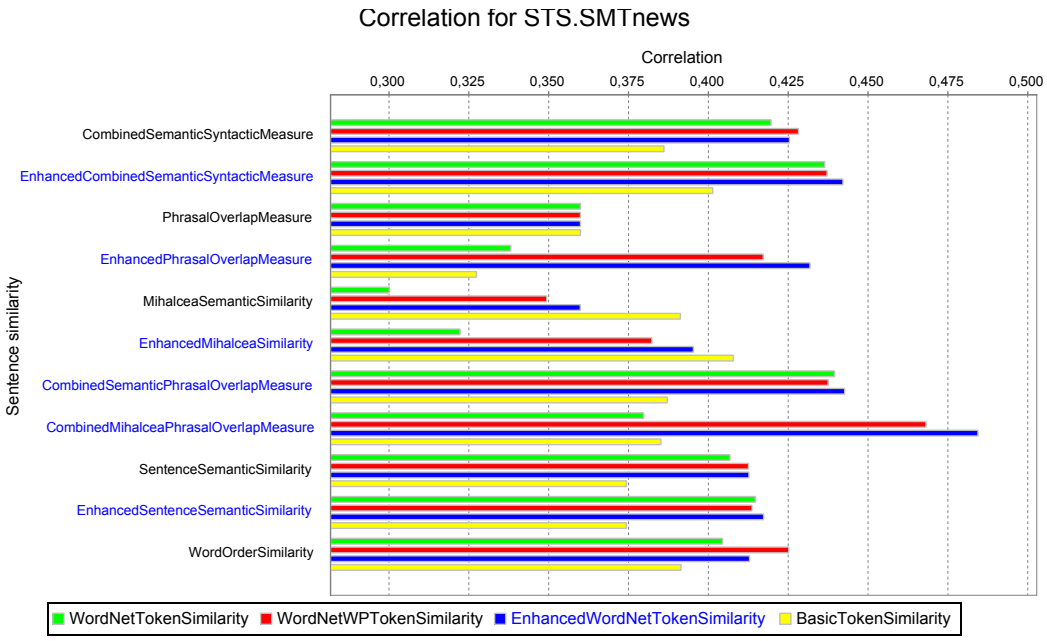

Fig. 6.9: Correlation between STS.SMTnews and the results of the examined similarity methods.

Participant	Correlation					
	ALL	MSRpar	MSRvid	SMTeur	OnWN	SMTnews
First place	0.8239	0.6830	0.8739	0.5280	0.6641	0.4937
Our result	0.4594	0.2330	0.4666	0.3483	0.4507	0.4844
Baseline	0.3110	0.4334	0.2996	0.4542	0.5864	0.3908

Tab. 6.2: STS task's results

on the 70th place out of 90 participants. On STS.SMTnews our results are basically equivalent. In comparison to the baseline our results are better on STS.ALL, STS.MSRvid and STS.SMTnews data sets.

We computed the percentage difference of our best result in comparison to the first place and baseline. It is shown on table 6.3. As you can see our result for STS.SMTnews is only 1.88% inferior to the result of the winner. The STS.MSRpar, STS.SMTeur data sets are the weakness of our similarity measure otherwise we achieved at least the correlation of 0.4507. Our similarity measure is better then the baseline by 23.95% on the STS.SMTnews data set and by 55.75% on the STS.MSRvid data set. The overall result for STS.ALL data set is better than the baseline by 47.72%. That is quite good considering that our system computes results within two minutes (Intel Core i5-430M, 4GB RAM, JDK 1.6.0_20, windows 7) and doesn't perform

deep parsing process such as Syntax-based measure for short-text semantic similarity (SyMSS section 2.5.2).

Description	STS data set					
	ALL	MSRpar	MSRvid	SMTeur	OnWN	SMTnews
Percentage difference of our result compared to the first place	-44.24%	-65.89%	-46.60%	-34.04%	-32.13%	-1.88%
Percentage difference of our result compared to the baseline	47.72%	-46.25%	55.75%	-23.33%	-23.14%	23.95%

Tab. 6.3: Percentage difference of STS task's results

7 Conclusion

We studied computer methods for sentence semantic similarity and the existing solutions with regard to the algorithms and suitability for determining similarity between sentences. We proposed six new sentence similarity measures and implemented them along with the selected existing methods. We subjected them to testing on two different data sets, the Microsoft Research paraphrase corpus and the Semantic Textual Similarity shared task.

We use different evaluation criteria on each data set. In the MSRP data set only two values are assigned to each sentence pair(semantically equivalent or semantically not equivalent), thus we chose accuracy (section 2.7.1) as the main evaluation criteria. Rejection rate (section 2.7.2) and acceptance rate (section 2.7.3) are additional metrics used to evaluate the MSRP data set. For the STS data set the evaluation criteria was given. The STS shared task has Pearson's correlation (section 2.7.4) as its official score, thus we chose correlation as well. Because the effectiveness of sentence similarity measures strongly depends on the examined data, we analyzed the data sets in section 6.1.

The evaluation (in section 6) demonstrates that the proposed semantic similarity measures are equivalent or even better than the state-of-the-art measures.

Our proposed sentence similarity method (*Combined Mihalcea Phrasal Overlap Measure* with *Enhanced WordNet Token Similarity*) is better than the baseline of the STS shared tasks by 47.72%. On the STS.SMTnews data set our result is only 1.88% inferior to the result of the winner of the task. In the ranking of the STS shared task we would be on the 70th place out of 90 participants.

Future work would include the construction of sentence similarity measures which include deep parsing process. We would exchange WordNet for another knowledge base that has better coverage of words and part of speech classes.

Bibliography

[1] Java API for WordNet Searching. URL: <http://lyle.smu.edu/~tspell/jaws/>, [online], cit. 2012-05-08.

[2] Java Excel API. URL: <http://jexcelapi.sourceforge.net/>, [online], cit. 2012-05-08.

[3] Java WordNet Interface. URL: <http://projects.csail.mit.edu/jwi/>, [online], cit. 2012-05-08.

[4] Java WordNet Library. URL: <http://sourceforge.net/projects/jwordnet/>, [online], cit. 2012-04-18.

[5] JFreeChart. URL: <http://www.jfree.org/jfreechart/>, [online], cit. 2012-04-17.

[6] *Semantic Textual Similarity (STS) shared task*, 2012. URL: <http://www.cs.york.ac.uk/semeval-2012/task6/>, [online], cit. 2012-04-18.

[7] *Semantic Textual Similarity (STS) shared task results*, 2012. URL: <http://www.cs.york.ac.uk/semeval-2012/task6/index.php?id=results-update>, [online], cit. 2012-05-7.

[8] P. Achananuparp, X. Hu, and X. Shen. The Evaluation of Sentence Similarity Measures. In *Proceedings of the 10th international conference on Data Warehousing and Knowledge Discovery*. Springer-Verlag, 2008.

[9] L. Azzopardi, M. Girolami, and M. Crowe. Probabilistic Hyperspace Analogue to Language. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, 2005.

[10] M.-C. de Marneffe, B. MacCartney, and C. D. Manning. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings*

of the IEEE / ACL 2006 Workshop on Spoken Language Technology. The Stanford Natural Language Processing Group, 2006.

[11] W. Dolan, C. Quirk, and C. Brockett. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *In Proceedings of the 20th International Conference on Computational Linguistics*, 2004.

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

[13] W. K. Gad and M. S. Kamel. New Semantic Similarity Based Model for Text Clustering Using Extended Gloss Overlaps. In *Proceedings of the 6th International Conference on Machine Learning and Data Mining in Pattern Recognition*, MLDM '09. Springer-Verlag, 2009.

[14] D. Higgins and J. Burstein. Sentence similarity measures for essay coherence. 2007.

[15] E. H. Hovy, M. P. Marcus, M. Palmer, L. A. Ramshaw, and R. M. Weischedel. Ontonotes: The 90 In *HLT-NAACL*. The Association for Computational Linguistics, 2006.

[16] D. Húsek, J. Pokorný, V. Snášel, and H. Řezanková. Metody vyhledávání v rozsáhlých kolekcích dat. In *Datakon 2003*, Brno, Czech Republic, 2003.

[17] P. Humpál. *Vyhledávání textu v indexovaných zdrojích*, Diplom thesis, 2011. University of West Bohemia Faculty of Applied Sciences.

[18] JUnit.org. *JUnit Specification*, 2010. URL: <`http://kentbeck.github.com/junit/javadoc/latest/`>, [online], cit. 2010-04-26.

[19] T. K. Landauer, P. W. Foltz, and D. Laham. An Introduction to Latent Semantic Analysis. 1998.

[20] Y. Li, D. McLean, Z. A. Bandar, J. D. O'Shea, and K. Crockett. Sentence Similarity Based on Semantic Nets and Corpus Statistics. volume 18, Aug. 2006.

[21] B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2nd edition, 2011.

[22] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[23] R. Mihalcea, C. Corley, and C. Strapparava. Corpus-based and Knowledge-based Measures of Text Semantic Similarity. In *Proceedings of the 21st national conference on Artificial intelligence - Volume 1*. AAAI Press, 2006.

[24] J. Oliva, J. I. Serrano, M. D. del Castillo, and A. Iglesias. SyMSS: A syntax-based measure for short-text semantic similarity. *Data Knowl. Eng.*, 70(4):390–405, 2011.

[25] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 252–259. Association for Computational Linguistics, 2003.

[26] Z. Wu and M. S. Palmer. Verb Semantics and Lexical Selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138, 1994.