Dependency vs. Constituent Based Syntactic N-Grams in Text Similarity Measures for Paraphrase Recognition

Hiram Calvo, Andrea Segura-Olivares, and Alejandro García

Centro de Investigación en Computación (CIC),
Instituto Politécnico Nacional (IPN),
Mexico City, Mexico

hcalvo@cic.ipn.mx, {msegura_b12, igarcia_b12}@sagitario.cic.ipn.mx

Abstract. Paraphrase recognition consists in detecting if an expression restated as another expression contains the same information. Traditionally, for solving this problem, several lexical, syntactic and semantic based techniques are used. For measuring word overlapping, most of the works use n-grams; however syntactic n-grams have been scantily explored. We propose using syntactic dependency and constituent n-grams combined with common NLP techniques such as *stemming, synonym detection, similarity measures*, and *linear combination* and a similarity matrix built in turn from syntactic n-grams. We measure and compare the performance of our system by using the Microsoft Research Paraphrase Corpus. An in-depth research is presented in order to present the strengths and weaknesses of each approach, as well as a common error analysis section. Our main motivation was to determine which syntactic approach had a better performance for this task: syntactic dependency n-grams, or syntactic constituent n-grams. We compare too both approaches with traditional n-grams and state-of-the-art systems.

Keywords. Paraphrase recognition, Microsoft Research paraphrase corpus, similarity measures, syntactic n-grams, constituent analysis, dependency analysis.

1 Introduction

It is known that syntactic n-grams present an advantage over traditional n-grams, since they are based in syntactic relationships of words, so that each word is associated with their "real" neighbor, ignoring arbitrarinesses that could be present at a surface level [18, 22].

Consider the expression *"the small funny dog barks"* and its syntactic dependency analysis tree shown in Figure 1. Its corresponding bigrams are listed in Table 1, where we can see that some bigrams have no meaning, such as *"the small"* and *"small funny"*.

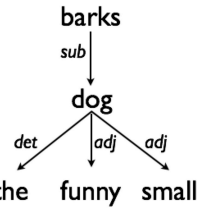

Fig. 1. Example of dependency parse tree

Table 1. Comparison between traditional and syntactic bigrams

Traditional bigrams	Syntactic bigrams
the small	barks dog
small funny	dog the
funny dog	dog funny
dog barks	dog small

Our intuition, based on recent studies applied to related areas [23, 20], is that syntactic n-grams can help to improve precision for paraphrase recognition, since they consider not only the expressions' words, but also their part of speech. A disadvantage of syntactic n-grams might be the need of a parser, which can be slow and may not be available for all languages, so that the benefits of using this additional resource should be clear. In this work we present an in-depth research in order to present the strengths and weaknesses of each approach. Our main motivation is to determine which syn-

ISSN 1405-5546
DOI: 10.13053/CyS-18-3-2044

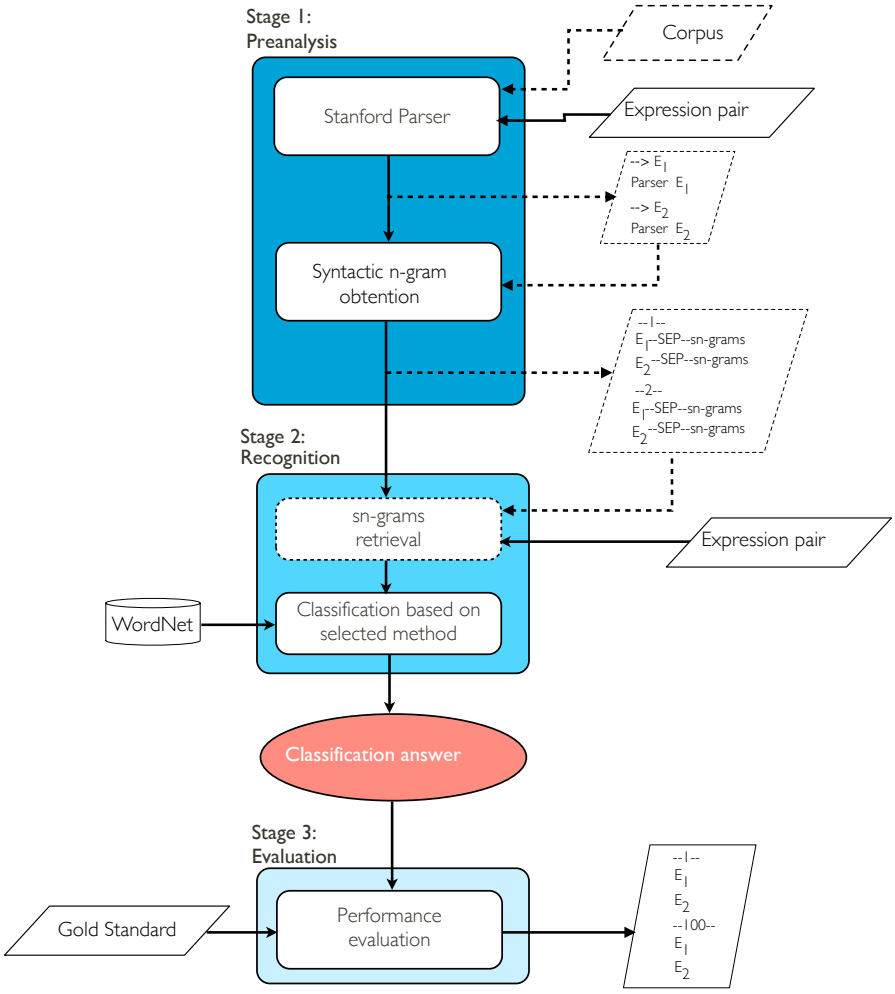

Fig. 2. Proposed architecture for paraphrase recognition

tactic approach performs the best for recognizing paraphrases: syntactic dependency n-grams, or syntactic constituent n-grams.

This paper is organized as follows. Section 2 gives details about our proposal, along with the resources used for evaluation: the Microsoft Research Paraphrase Corpus (Section 2.1), Syntax analyzers (Section 2.2), and details about the auxiliary NLP techniques used in this work (see Section 2.3). In Section 3 we present experiments and results of our proposal. First, we present results about threshold adjustments (Section 3.1); then, we present results for syntactic dependency n-grams (Section 3.2), and syntactic constituent n-grams (Section 3.3). We will consider the special case of syntactic unigrams in Section 3.4; a comparison of syntactic dependency n-grams vs. syntactic constituent n-grams (Section 3.5), and finally a comparison between the proposed methods and traditional n-grams (Section 3.6).

We devote a special section to error analysis (Section 4) and then we compare our approach with the existing methods in the State of the Art in Section 5. Finally, we draw our conclusions in Section 6.

2 Paraphrase Recognition using Syntactic N-grams

The general architecture proposed for Paraphrase recognition is shown in Fig. 2. This can be divided in two stages: (I) syntactic preanalysis, where syntactic n-grams are obtained, as described in Section 2.2; and (II) the recognition step, where the syntactic n-grams corresponding to the pair of received expressions are used by a classification module that decides if the pair of expressions is a paraphrase or not. As a third stage, each recognition method is evaluated against the Gold Standard proposed by the Microsoft Research Corpus.

2.1 The Microsoft Research Paraphrase Corpus (MSRP)

The Microsoft Research Paraphrase Corpus (MSRP) is an standard used for evaluating paraphrase recognition methods [3]. We observed, however, that the *MSRP* is unbalanced, since the 67.5% and 66.5% of all pairs are positive for the training and test sets, respectively.

Because of this, we experienced some problems, such as comparing against a baseline. Usually for this kind of system, the baseline is computed by choosing always the same answer. For this particular corpus, if each presented pair is chosen as true, F measure results in 79.9% which is a relatively high value to be considered as a lower minimum. Table 2 shows the baseline results for the test set.

Table 2. Results obtained with the baseline system

Accuracy	Precision	Recall	F-Measure
66.5%	66.5%	100%	79.9%

The reason why the Microsoft Paraphrase Recognition corpus is unbalanced might be the way in which it was created, since initially 20,574 pairs of possible paraphrases were available, from which 5,801 were randomly selected without a specific balance criterion for positive and negative cases.

2.2 Syntactic Preanalysis

In this work we propose using the Stanford parser, originally created by Dan Klein and Christopher Manning. There are many other parsers such as MiniPar [10], Collins, Charniak, etc.; it is not easy to say which one is the best, however Stanford parser has been found to have an unbiased performance with regard to recall and F-measure [25]; in addition, it yields constituent and dependency parses as well.

An example of the input expressions found in the syntactic preanalysis is the following:

```
Amrozi accused his brother, whom
he called "the witness", of deliberately
distorting his evidence.

Referring to him as only "the witness",
Amrozi accused his brother of
deliberately distorting his evidence.
```

For these, the following output is obtained by using the Stanford Syntactic Parser:

```
———>Amrozi accused his brother, whom he
called "the witness", of
deliberately distorting his evidence.

accused-2:Amrozi-1<-->ROOT-0:accused-2
<-->brother-4:his-3<-->accused-2:
brother-4<-->called-8:brother-4<-->
called-8:he-7<-->brother-4:called-8
<-->witness-11:the-10<-->called-8:
witness-11<-->distorting-16:
deliberately-15<-->brother-4:
distorting-16<-->evidence-18:his-17
<-->distorting-16:evidence-18

———>Referring to him as only "the
witness", Amrozi accused his brother
of deliberately distorting his evidence.
accused-12:Referring-1<-->Referring-1:
him-3<-->witness-8:only-5<-->witness-8:
the-7<-->Referring-1:witness-8<-->
accused-12:Amrozi-11<-->ROOT-0:accused-12
<-->brother-14:his-13<-->accused-12:
brother-14<-->distorting-17:
deliberately-16<-->brother-14:
distorting-17<-->evidence-19:
his-18<-->distorting-17:evidence-19
```

Subsequently, the syntactic n-grams are extracted, and they are stored in a database as follows:

```
—1—
702876——SEPARATOR—Amrozi accused his
brother, whom he called "the witness",
of deliberately distorting his evidence.
—SEPARATOR—accused:amrozi<—>root:
accused<—>brother:his<—>accused:brother
<—>called:brother<—>called:he<—>
brother:called<—>witness:the<—>called:
witness<—>distorting:deliberately<—>
brother:distorting<—>evidence:his<—>
distorting:evidence702977—SEPARATOR—
Referring to him as only "the witness",
Amrozi accused his brother of deliberately
distorting his evidence. —SEPARATOR—
accused:referring<—>referring:him<—>
witness:only<—>witness:the<—>referring:
witness<—>accused:amrozi<—>root:accused
<—>brother:his<—>accused:brother<—>
distorting:deliberately<—>brother:
distorting<—>evidence:his<—>
distorting:evidence
```

The word **–SEPARATOR–** separates the elements of each expression; firstly the expression identifier (702876 and 702977 for this example), the expression itself, and finally the syntactic n-grams corresponding to the expression. In this example, bigrams. These files were generated for trigrams, and tetragrams for the dependency and constituent analysis as well.

2.3 Auxiliary NLP Techniques

For this work, several NLP techniques were used in conjunction with syntactic n-grams. We will describe them briefly in the next sections, as well as the syntactic n-grams extraction process itself.

In general the process is as follows:

1. Analize an expression with the Stanford parser.
2. Use the dependency or constituent relationships obtained in the previous step to form syntactic bigrams, trigrams and tetragrams.

For more details on the syntactic n-grams extraction, please refer to [17].

Table 3. Example of syntactic bigrams with synonyms

Synonyms(education, disaster)
(education, disaster)
(instruction, disaster)
(education, catastrophe)
(instruction, catastrophe)
Synonyms (school, accident)
(school, accident)
(schoolhouse, accident)
(school, stroke)
(schoolhouse, stroke)

2.3.1 Negation

A basic negation scheme was applied to the input string before parsing. For example, for words such as **can't**, **wouldn't**, **not**, **isn't**, **ain't**, etc., the subsequent words are *negated* by adding **not** to each of them until finding a period, colon or semicolon. For example:

Original string: *Liquid water can not exist on mars.*

String with negation: *Liquid water can not exist not on not mars.*

2.3.2 Synonym Detection

This technique is important because in many cases paraphrase pairs are created by substituting some words with their synonyms. We expanded each syntactic n-gram with the nearest synonym of each of its members. For example, for bigrams *(car, red)* and *(NN, car)*:

$$synonyms(car, red) \begin{cases} (car, red) \\ \textbf{\textit{(auto, red)}} \\ (car, redness) \\ (auto, redness) \end{cases}$$

$$synonyms(NN, car) \begin{cases} (NN, car) \\ \textbf{\textit{(NN, auto)}} \end{cases}$$

Each syntactic n-gram produces 2^n new pairs, where n is 2 for bigrams, 3 for trigrams and 4 for tetragrams; see Table 3.

2.3.3 Lin Similarity Measure

When two words are related, but they are not necesarily synonyms, a distributional similarity measure such as the Lin similarity measure [9] is convenient. This measure is based on the WordNet hierarchy. For example, consider the syntactic bigrams *(education, disaster)* and *(school, accident)*: there would be no overlap, even if we consider their synonyms.

With the Lin similarity measure, we have for this example that the similarity between *education* and *school* is 0.84; between *disaster* and *accident* is 0.88. If we consider a similarity binarization threshold of 0.8 (empirically determined), then the pairs *(education, disaster)* and *(school, accident)* would be considered as equivalent.

3 Experiments and Results

3.1 Threshold Adjustment

Each one of the experiments is based on a threshold that can be of similarity or difference depending on the method used. We performed several tests with different thresholds ranging from 0.05 to 0.95 with an incremental step of 0.05. For the syntactic dependency analysis we used first the *MSRP training set* for each proposed combination for syntactic bigrams, trigrams and tetragrams. Then, we obtained the optimal threshold with regard to the F-measure, and we used it on the test set. In most cases, the threshold was selected so that the baseline system's performance was approached, by classifying all input pairs as true; however, this would not be an interesting value, because any pair would be classified as true in such combination; this is shown in Figure 3. No auxiliary techniques were used for such results. In that Figure can be also seen that baseline's performance could not be outperformed, since the maximum score was obtained with a threshold of 0.95.

Therefore, aiming to obtain a better threshold estimation, we experimented with a balanced training set, with the intuition that this could result in a better threshold that could outperform baseline's performance. In order to do this, we took all 1,323 false pairs in the training corpus, and then we randomly selected the same quantity of positive pairs from the same corpus. As a result, we obtained an optimal threshold for each combination of the syntactic n-grams. Additionaly, a fixed threshold (.085) was selected for comparison between syntactic bigrams, trigrams and tetragrams. By using the same threshold, they can be compared in the same conditions. Figure 4 shows the basic analysis' performance for syntactic dependency bigrams, using a balanced corpus, here we can see an optimal performance for a threshold of 0.85. This procedure was done for each one of the remaining combinations in order to obtain an optimal threshold for evaluation with the test set.

With regard to syntactic constituent analysis, finding the optimal threshold was done considering the unbalanced training corpus, finding the best value for each combination for later evaluation with the test set. As well as with the dependency syntax analysis, a fixed threshold was set (0.55) for comparison between different syntactic constituent n-grams. Figure 5 shows results for the basic syntactic constituent analysis with bigrams, using the unbalanced corpus, from which it is possible to find the highest performance when using a threshold of 0.55.

3.2 Experiments with Syntactic N-grams using Dependency Analysis

In this section, we describe the results of our experiments for paraphrase recognition. We will compare syntactic dependency bigrams, trigrams, and tetragrams, using a fixed thershold of 0.85 for all three diferent syntactic n-grams, and the combination of different NLP techniques. Additionaly, we will present results using the linear combination and similarity matrix, aiming to improve the syntactic dependency n-grams' performance.

3.2.1 Syntactic Bigrams

Next we present results corresponding to the use of several NLP techniques along with syntactic dependency bigrams with the train and test sets respectively in tables 4 and 5.

We can see that using both **stemming** and **synonyms** ("O" key) has the best score for the training

ISSN 1405-5546
DOI: 10.13053/CyS-18-3-2044

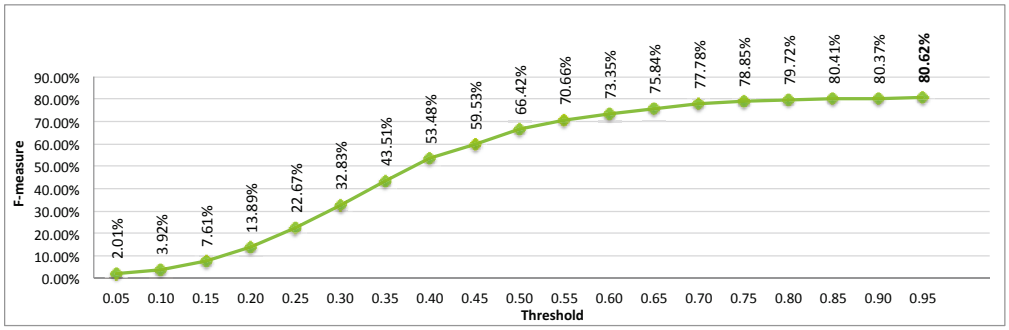

Fig. 3. Performance in unbalanced corpus for dependency n-grams

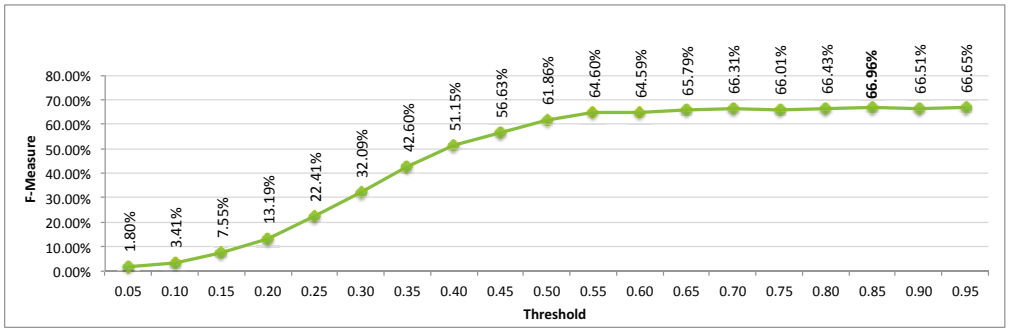

Fig. 4. Performance of syntactic dependency analysis with a balanced corpus

and test sets; on the other hand, for the test set, using **synonyms** itself ("D" key) achieves the same F-measure score than the previous combination, with a threshold of 0.85 for both cases.

3.2.2 Syntactic Trigrams

Here we present the results corresponding to the syntactic dependency trigrams for the training and test sets in tables 6 and 7 respectively.

From previous tables, we can see that the combination of the **synonyms** and **stopwords removal** techniques ("L" key), achieves the best F-measure for the training set when using a threshold of 0.95. It is worth noting that in this case that combination and threshold practically correspond to the baseline. This is discussed in more detail in Section 3.2.4. With the test set, the combination of **stemming** and **Lin's similarity** measure, **negation** and **stopwords** removal ("S" key), has the best score with a threshold of 0.90.

3.2.3 Syntactic Tetragrams

Finaly, for dependencies, tables 8 and 9 show results of experiments with the training and test sets respectively.

We can see from previous tables, that the combination of **stemming**, **Lin's similarity** measure, and **stopwords** removal ("N" key), yields the best F-score for the training set, using a threshold of 0.95. For the test set, Combining **Lin's similarity** measure and **stopwords** removal ("K" key) yields the best F-measure with the same threshold. It is worth noting that in both cases baseline is not outperformed. See Section 2.1.

3.2.4 Analysis and Evaluation for Syntactic Dependency N-grams

First we analyze the optimal threshold for each studied syntactic dependency N-gram, so that we

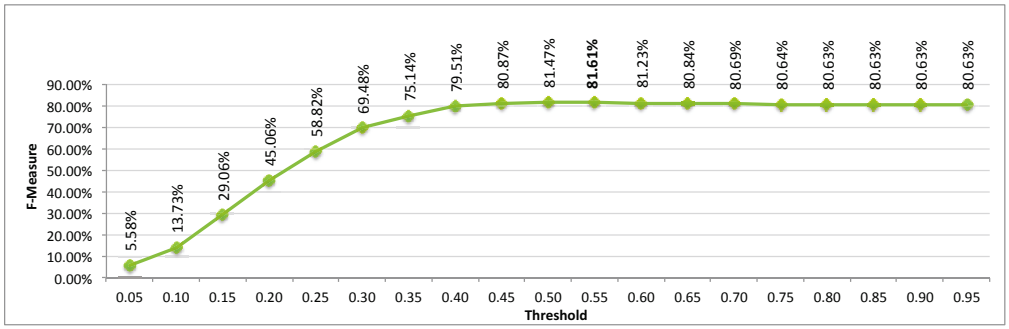

Fig. 5. Performance of syntactic constituent analysis with an unbalanced corpus

Table 4. Experiments with the training set for syntactic dependency bigrams

Key	Stemming	Lin similarity measure	Synonyms	Negation	Stop words removal	Accuracy	Precision	Recall	F-Measure	Threshold
A						68.00%	68.55%	97.23%	80.41%	0.85
B	✓					68.17%	68.59%	97.56%	80.55%	0.85
C		✓				69.23%	72.62%	87.39%	79.32%	0.65
D			✓			68.10%	68.55%	97.49%	80.50%	0.85
E				✓		67.93%	68.62%	96.76%	80.30%	0.85
F					✓	67.86%	68.57%	96.76%	80.26%	0.85
G	✓			✓		68.15%	68.66%	97.20%	80.48%	0.85
H		✓		✓		69.16%	72.88%	86.52%	79.12%	0.65
I			✓	✓		68.05%	68.63%	97.05%	80.40%	0.85
J	✓				✓	67.98%	68.57%	97.09%	80.37%	0.85
K		✓			✓	68.44%	70.91%	90.33%	79.45%	0.70
L			✓		✓	67.71%	69.98%	91.39%	79.26%	0.75
M	✓	✓				69.43%	72.16%	89.10%	79.74%	0.65
N	✓	✓			✓	68.71%	73.43%	84.12%	78.41%	0.60
O	✓		✓			**68.25%**	**68.60%**	**97.71%**	**80.61%**	**0.85**
P	✓		✓		✓	67.88%	69.97%	91.86%	79.44%	0.75
Q	✓			✓	✓	67.71%	70.17%	90.77%	79.15%	0.75
R	✓	✓		✓		69.43%	72.44%	88.33%	79.60%	0.65
S	✓	✓		✓	✓	69.08%	72.60%	87.10%	79.19%	0.65
T	✓		✓	✓		68.22%	68.68%	97.34%	80.54%	0.85
U	✓		✓	✓	✓	67.86%	70.17%	91.17%	79.30%	0.75
Baseline						67.54%	67.54%	100.00%	80.62%	–

Table 5. Experiments with the test set for syntactic dependency bigrams

Key	Stemming	Lin similarity measure	Synonyms	Negation	Stop words removal	Accuracy	Precision	Recall	F-Measure	Threshold
A						68.63%	68.56%	97.55%	80.53%	0.85
B	✓					68.57%	68.45%	97.82%	80.54%	0.85
C		✓				69.50%	72.91%	86.13%	78.97%	0.65
D			✓			68.69%	68.54%	97.82%	80.60%	0.85
E				✓		68.69%	68.67%	97.29%	80.51%	0.85
F					✓	67.88%	68.20%	96.86%	80.04%	0.85
G	✓			✓		68.63%	68.56%	97.55%	80.53%	0.85
H		✓		✓		69.56%	73.45%	84.91%	78.77%	0.65
I			✓	✓		68.75%	68.65%	97.55%	80.59%	0.85
J	✓				✓	68.11%	68.23%	97.38%	80.24%	0.85
K		✓			✓	68.34%	70.53%	89.97%	79.08%	0.70
L			✓		✓	68.00%	70.08%	90.49%	78.99%	0.75
M	✓	✓				69.56%	72.11%	88.40%	79.43%	0.65
N	✓	✓			✓	69.10%	73.68%	83.26%	78.18%	0.60
O	✓		✓			**68.63%**	**68.45%**	**97.99%**	**80.60%**	**0.85**
P	✓		✓		✓	68.40%	70.12%	91.45%	79.37%	0.75
Q	✓			✓	✓	68.34%	70.53%	89.97%	79.08%	0.75
R	✓	✓		✓		69.68%	72.64%	87.27%	79.28%	0.65
S	✓	✓		✓	✓	68.28%	72.02%	85.52%	78.19%	0.65
T	✓		✓	✓		68.69%	68.56%	97.73%	80.58%	0.85
U	✓		✓	✓	✓	68.28%	70.46%	90.06%	79.06%	0.75
Baseline						66.49%	66.49%	100.00%	79.87%	–

are able to know the highest score that can be obtained with each one of them, along with the combination of NLP techniques. For bigrams we obtained an F-measure of **80.60%** using **synonyms** ("D" key) and the combination of **stemming** and **synonyms** ("O" key).

The difference between them is that the first one has better accuracy and precision, but a lower recall. However, we selected "O" as the best technique since it improves performance in the training set as well. In general, these combinations were the ones that achieved the best performance in F-measure for the three studied syntactic dependency n-grams. Performance for each proposed techniques when using their optimal threshold is shown in Figure 6 along with their comparison with the baseline system.

As can be seen, only in 10 of 21 tests the baseline is outperformed. The worst combination seems to be using stemming, Lin's similarity measure, and stopwords removal ("N" key) with an F-measure of 78.18%.

For syntactic dependency trigrams, even less NLP techniques combinations improve scores with

Table 6. Experiments with the training set for syntactic dependency trigrams

Key	Stemming	Lin similarity measure	Synonyms	Negation	Stop words removal	Accuracy	Precision	Recall	F-Measure	Threshold
A						67.07%	67.96%	96.94%	79.91%	0.95
B	✓					67.39%	71.44%	86.16%	78.11%	0.90
C		✓				68.59%	70.83%	90.95%	79.64%	0.90
D			✓			67.12%	67.98%	97.02%	79.94%	0.95
E				✓		66.90%	67.99%	96.36%	79.72%	0.95
F					✓	67.76%	67.84%	99.34%	80.63%	0.95
G	✓			✓		67.07%	68.02%	96.73%	79.87%	0.95
H		✓		✓		68.67%	71.07%	90.41%	79.58%	0.90
I			✓	✓		66.60%	71.48%	84.12%	77.29%	0.90
J	✓				✓	68.03%	70.58%	90.30%	79.23%	0.90
K		✓			✓	68.81%	70.27%	93.28%	80.16%	0.90
L			✓		✓	**67.78%**	**67.85%**	**99.38%**	**80.64%**	**0.95**
M	✓	✓				68.69%	70.59%	91.93%	79.86%	0.90
N	✓	✓			✓	68.76%	70.05%	93.89%	80.24%	0.90
O	✓		✓			67.51%	71.46%	86.41%	78.23%	0.90
P	✓		✓		✓	68.15%	70.61%	90.51%	79.33%	0.90
Q	✓			✓	✓	67.78%	67.89%	99.23%	80.62%	0.95
R	✓	✓		✓		68.79%	70.84%	91.42%	79.82%	0.90
S	✓	✓		✓	✓	68.69%	70.18%	93.28%	80.09%	0.90
T	✓		✓	✓		67.39%	71.64%	85.61%	78.00%	0.90
U	✓		✓	✓	✓	68.00%	70.76%	89.68%	79.10%	0.90
Baseline						67.54%	67.54%	100.00%	80.62%	–

regard to the baseline, being only 3 combinations in this case. The best one of them has an F-measure of **80.06%** with the combination of **stemming**, **Lin's similarity** measure, **negation**, and **stopwords** removal ("S" key). This clashes in turn with the low performance of the same combination with syntactic bigrams.

Regarding the training set, the combination of **synonyms** and **stopwords** ("L" key), had an optimal performance. Since in "S" and "L" the technique of stopwords removal is present, we conclude this technique is useful for syntactic dependency trigrams. Performance per combination of techniques is shown in Figure 7. The best combination is highlighted in bold, and a comparison against baseline is shown. In contrast, the worst performance corresponded to the combination of **synonyms** and **negation**. ("I" key) with an F-score of 77.08%.

Finally, results for syntactic dependency tetragrams are shown in Figure 8. In this figure, we can see that no combination was able to outperform the baseline, in terms of the F-measure. The best performance was obtained with the combination of

Table 7. Experiments with the test set for syntactic dependency trigrams

Key	Stemming	Lin similarity measure	Synonyms	Negation	Stop words removal	Accuracy	Precision	Recall	F-Measure	Threshold
A						66.55%	67.12%	97.38%	79.47%	0.95
B	✓					67.47%	71.17%	85.87%	77.83%	0.90
C		✓				68.69%	70.54%	90.84%	79.42%	0.90
D			✓			66.60%	67.12%	97.55%	79.53%	0.95
E				✓		66.26%	67.04%	96.86%	79.24%	0.95
F					✓	66.55%	66.72%	99.12%	79.76%	0.95
G	✓			✓		66.43%	67.08%	97.21%	79.38%	0.95
H		✓		✓		68.69%	70.74%	90.23%	79.31%	0.90
I			✓	✓		66.89%	71.42%	83.69%	77.07%	0.90
J	✓				✓	68.34%	70.31%	90.67%	79.20%	0.90
K		✓			✓	68.86%	69.80%	93.72%	80.01%	0.90
L			✓		✓	66.49%	66.68%	99.12%	79.73%	0.95
M	✓	✓				68.28%	70.00%	91.54%	79.33%	0.90
N	✓	✓			✓	68.75%	69.58%	94.15%	80.02%	0.90
O	✓		✓			67.53%	71.19%	85.96%	77.88%	0.90
P	✓		✓		✓	68.40%	70.33%	90.75%	79.25%	0.90
Q	✓			✓	✓	66.37%	66.64%	98.95%	79.64%	0.95
R	✓	✓		✓		68.28%	70.18%	90.93%	79.22%	0.90
S	✓	✓		✓	✓	**68.98%**	**69.92%**	**93.63%**	**80.05%**	**0.90**
T	✓		✓	✓		67.47%	71.44%	85.09%	77.67%	0.90
U	✓		✓	✓	✓	68.63%	70.75%	90.06%	79.24%	0.90
Baseline						66.49%	66.49%	100.00%	79.87%	–

Lin's similarity measure and **stopwords** removal ("K" key) with a score of **79.54%**.

For the training set, the best combination was obtained with the **stemming**, **Lin's similarity** and **stopwords** removal techniques ("N" key) with an F-measure of 80.05%, so that at this point **Lin's similarity** measure and **stopwords** removal appear to be useful techniques to apply when using syntactic dependency tetragrams.

Finally, for syntactic dependency tetragrams, the worst combination used **negation** only ("I" key), with an F-measure of 78.07%.

To summarize, in Table 10, the best and worst syntactic dependency N-gram combinations' performance with the test set are shown. It is worth noting that each NLP technique can provide a different support depending on the techniques themselves and the threshold itself.

For example, using synonyms is the best for bigrams, but when used in combination with negation for trigrams, it becomes the worst combination.

We have shown so far the best and worst combinations for syntactic dependency bigrams, trigrams and tetragrams considering the best threshold for

Table 8. Experiments with the training set for syntactic dependency tetragrams

Key	Stemming	Lin similarity measure	Synonyms	Negation	Stop words removal	Accuracy	Precision	Recall	F-Measure	Threshold
A						65.26%	67.61%	93.20%	78.37%	0.95
B	✓					65.55%	67.67%	93.82%	78.63%	0.95
C		✓				66.46%	67.85%	95.67%	79.39%	0.95
D			✓			65.43%	67.67%	93.46%	78.50%	0.95
E				✓		65.08%	67.69%	92.40%	78.14%	0.95
F					✓	66.58%	67.47%	97.52%	79.76%	0.95
G	✓			✓		65.38%	67.74%	93.06%	78.40%	0.95
H		✓		✓		66.48%	67.99%	95.16%	79.32%	0.95
I			✓	✓		65.23%	67.74%	92.62%	78.25%	0.95
J	✓				✓	66.68%	67.51%	97.67%	79.83%	0.95
K		✓			✓	66.87%	67.55%	98.03%	79.99%	0.95
L			✓		✓	66.63%	67.49%	97.60%	79.80%	0.95
M	✓	✓				66.60%	67.86%	96.04%	79.53%	0.95
N	✓	✓			✓	**66.95%**	**67.56%**	**98.22%**	**80.05%**	**0.95**
O	✓		✓			65.60%	67.68%	93.89%	78.66%	0.95
P	✓		✓		✓	66.73%	67.52%	97.74%	79.87%	0.95
Q	✓			✓	✓	66.58%	67.56%	97.16%	79.70%	0.95
R	✓	✓		✓		66.63%	68.00%	95.56%	79.46%	0.95
S	✓	✓		✓	✓	66.92%	67.61%	97.92%	79.99%	0.95
T	✓		✓	✓		65.43%	67.75%	93.13%	78.44%	0.95
U	✓		✓	✓	✓	66.63%	67.58%	97.23%	79.74%	0.95
Baseline						67.54%	67.54%	100.00%	80.62%	–

each combination. Now we will present a comparison under the same conditions (threshold and NLP technique combination) so that we can perform a direct comparison between them. Figure 9 shows performance on the accuracy measure for bigrams, trigrams and tetragrams on the test set.

We can see here a favorable performance of bigrams, since in most combinations they outperform the baseline. Trigrams, however are less lucky (7 out of 21 combinations), and lastly, tetragrams are unable to outperform the baseline. For the case of **precision**, we have, however, that the worst performance is obtained by bigrams, being tetragrams the ones that have the best precision; see Figure 10.

Regarding **recall**, syntactic bigrams have the best recall, followed by trigrams, and tetragrams; see Figure 11.

Finally, compared by **F-measure**, syntactic bigrams have the best performance, mostly influenced by recall; see Figure 12.

Seen globally, it would seem that syntactic dependency bigrams outperform both trigrams and tetragrams; however, their precision is quite low.

Table 9. Experiments with the test set for syntactic dependency tetragrams

Key	Stemming	Lin similarity measure	Synonyms	Negation	Stop words removal	Accuracy	Precision	Recall	F-Measure	Threshold
A						65.44%	66.95%	94.85%	78.49%	0.95
B	✓					65.62%	66.97%	95.29%	78.66%	0.95
C		✓				65.97%	66.96%	96.33%	79.01%	0.95
D			✓			65.50%	66.95%	95.03%	78.55%	0.95
E				✓		64.92%	66.81%	93.89%	78.07%	0.95
F					✓	66.20%	66.74%	97.99%	79.40%	0.95
G	✓			✓		64.98%	66.76%	94.24%	78.16%	0.95
H		✓		✓		65.50%	66.84%	95.46%	78.63%	0.95
I			✓	✓		64.92%	66.79%	93.98%	78.08%	0.95
J	✓				✓	66.14%	66.68%	98.08%	79.39%	0.95
K		✓			✓	**66.31%**	**66.70%**	**98.51%**	**79.54%**	**0.95**
L			✓		✓	66.20%	66.72%	98.08%	79.42%	0.95
M	✓	✓				65.85%	66.86%	96.42%	78.97%	0.95
N	✓	✓			✓	66.20%	66.62%	98.51%	79.49%	0.95
O	✓		✓			65.68%	66.99%	95.37%	78.70%	0.95
P	✓		✓		✓	66.20%	66.70%	98.16%	79.43%	0.95
Q	✓			✓	✓	65.85%	66.60%	97.55%	79.16%	0.95
R	✓	✓		✓		65.33%	66.70%	95.55%	78.56%	0.95
S	✓	✓		✓	✓	65.85%	66.52%	97.90%	79.22%	0.95
T	✓		✓	✓		64.98%	66.76%	94.24%	78.16%	0.95
U	✓		✓	✓	✓	65.85%	66.60%	97.55%	79.16%	0.95
Baseline						66.49%	66.49%	100.00%	79.87%	–

Then, as in many cases, careful selection of the apropriate syntactic n-grams must be done, depending on the application.

3.2.5 Linear Combination for Syntactic Dependency N-grams

In this section, we show results of using linear combination for syntactic dependency n-grams. The linear combination method weights bigrams, trigrams and tetragrams by lambda values for each one of them, with

$$\lambda_4 + \lambda_3 + \lambda_2 = 1 \text{ and } \lambda_4, \lambda_3, \lambda_2 \geq 0$$

we experimented with all possible combinations of lambda values, considering steps of 0.1.

This yields 66 combinations, from which we tested 63—the remaining ones, in which two lambdas were zero, were discarded.

Those combinations were tested in the training and test sets without using any auxiliary NLP technique. For the test set the best values are shown in Table 11.

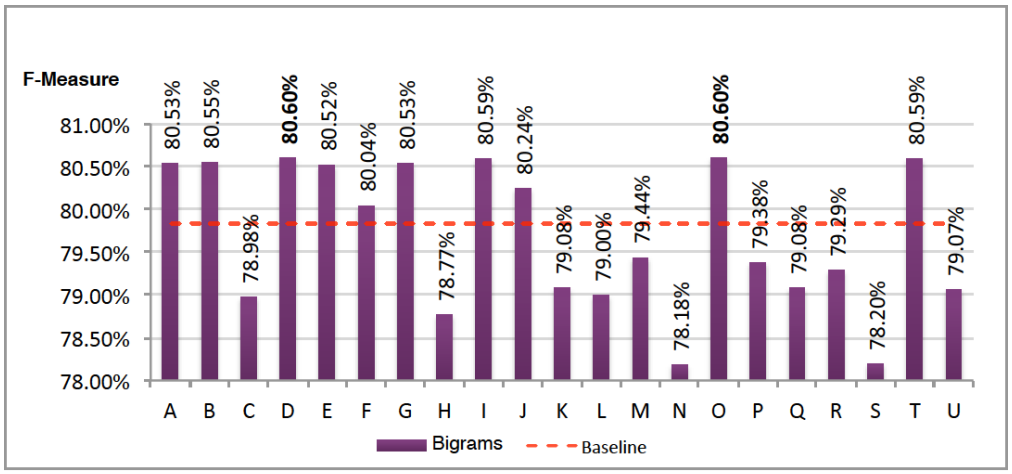

Fig. 6. Comparison between different NLP techniques used with syntactic dependency bigrams for the test set

Table 10. Summary of the best and the worst techniques for syntactic dependency n-grams

	Best combination	**Worst combination**
Syntactic Bigrams	Synonyms	Stemming + Lin's similarity measure + stopwords removal
Syntactic Trigrams	Stemming + Lin's similarity + negation + stopwords removal	Synonyms + negation
Syntactic Tetragrams	Lin's similarity + stopwords removal	Negation

Table 11. Best result for linear combination with no auxiliary NLP techniques

Lambda values	**Accuracy**	**Precision**	**Recall**	**F-measure**	**Threshold**
$\lambda_4 = 0.1$ $\lambda_3 = 0.0$ $\lambda_2 = 0.9$	68.69%	68.69%	97.21%	80.50%	0.10

3.2.6 Linear Combination Analysis

Additionally to the previous experiments, we experimented with using linear combination with stemming; see Figure 13.

We can see in Figure 13 that stemming yields a better performance in terms of recall and F-measure. This comparison was done using the best lambda values with and without stemming, respectively, in the training set.

In Table 12, the best 14 combinations of Lambda values are shown. We can see that in most of the combinations (13 out of 14), the greatest weight is given to the syntactic bigrams, i.e., λ_2. However, this occurs only for the F-measure; if precision was the score to optimize, the greatest weight should be given to the syntactic tetragrams.

Table 13 shows the best 5 results for precision, and their corresponding lambda weights.

Finally, we present a comparison between the highest scores obtained individually for bigrams, trigrams and tetragrams, versus the highest score obtained by linear combination with stemming; see Figure 14.

We can see that linear combination does not outperform syntactic bigrams and trigrams for accuracy and precision; however in Figure 15 we can see that the linear combination outperforms syntactic bigrams, trigrams and tetragrams in recall,

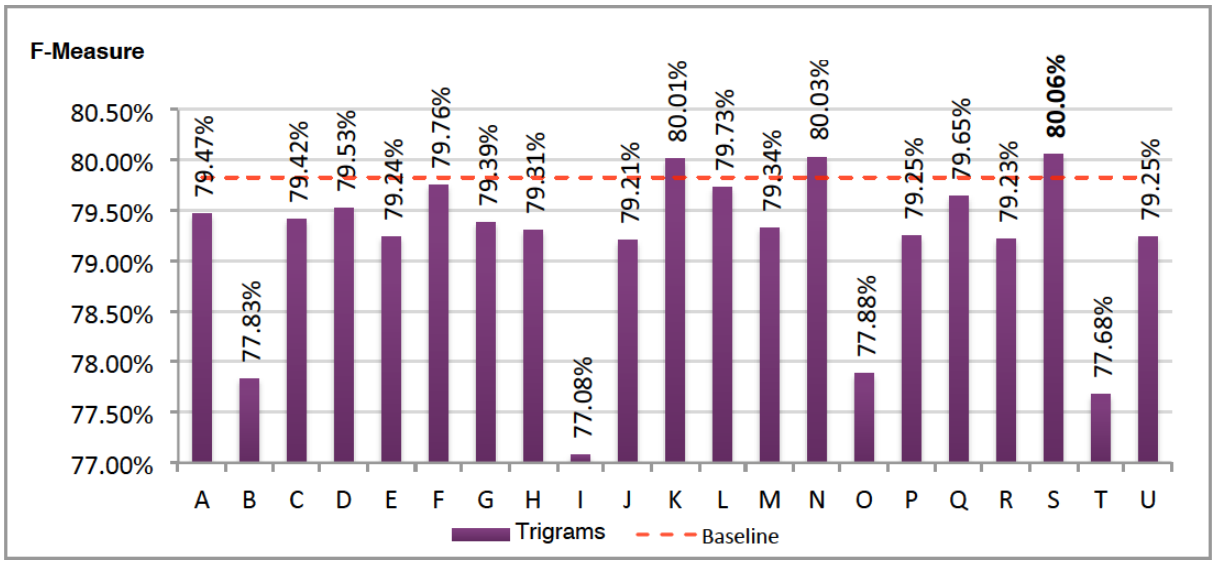

Fig. 7. Comparison between different NLP techniques used with syntactic dependency trigrams for the test set

Table 12. Best lambda values for linear combination in the test set

Key	λ_4	λ_3	λ_2
1	0.1	0.3	0.6
2	0.1	0.4	0.5
3	0.2	0.2	0.6
4	0.2	0.3	0.5
5	0.3	0.1	0.6
6	0.3	0.2	0.5
7	0.4	0.1	0.5
8	0.4	0.2	0.4
9	0.0	0.1	0.9
10	0.0	0.4	0.6
11	0.0	0.5	0.5
12	0.1	0.0	0.9
13	0.4	0.0	0.6
14	0.5	0.0	0.5

but not in the F-measure, being syntactic bigrams better.

3.2.7 Similarity Matrix with Syntactic Dependency N-grams

In order to explore further improvement of results using syntactic dependency n-grams, we experimented creating a similarity matrix based on syntactic dependency bigrams. The foundation of this method comes from [4]. In this work, they use a similarity matrix to calculate the similarity of two words by measuring the cosine angle for each word and its co-occurrent words within a window (it could be a paragraph or a sentence). Please refer to [4] for more details.

Table 13. Best 5 results for precision, and their corresponding lambda weights for the test set

Key	λ_4	λ_3	λ_2	Precision
1	0.5	0.5	0.0	72.44%
2	0.6	0.4	0.0	73.16%
3	0.7	0.3	0.0	73.17%
4	0.8	0.2	0.0	73.37%
5	0.9	0.1	0.0	74.20%

Table 14 shows results obtained for the training and test sets. We can see that according to the F-measure, the best threshold is 0.20, yielding an F-measure of 80.33%.

3.2.8 Similarity Matrix Analysis

Now we compare the best scores obtained by syntactic dependency bigrams, trigrams and tetragrams versus the Similarity Matrix method. Figure 16 shows that this latter method is in second

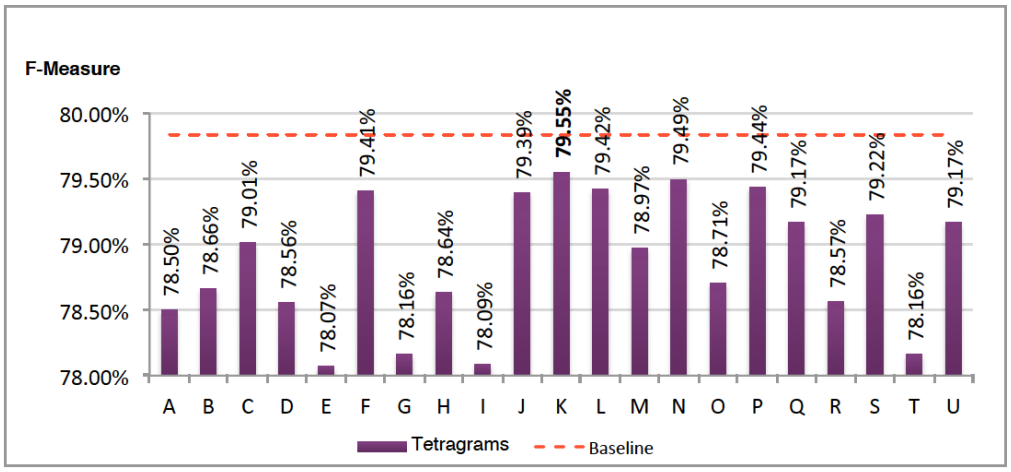

Fig. 8. Comparison between different NLP techniques used with syntactic dependency tetragrams for the test set

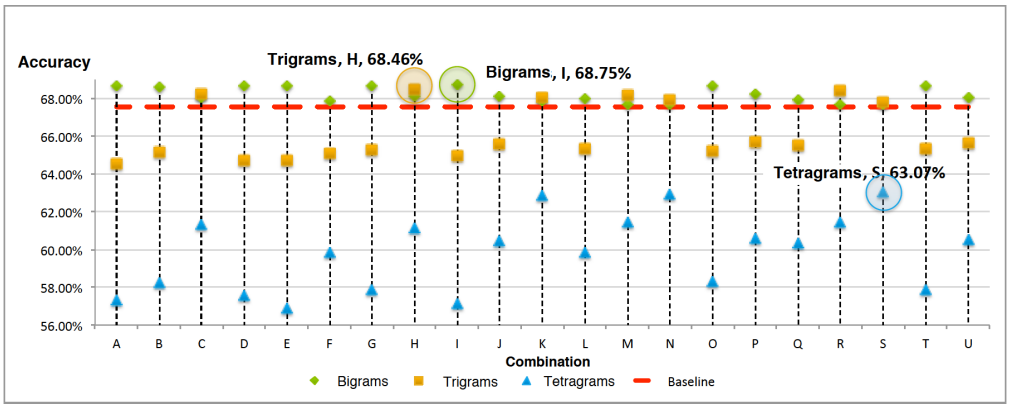

Fig. 9. Comparison of accuracy for syntactic dependency n-grams

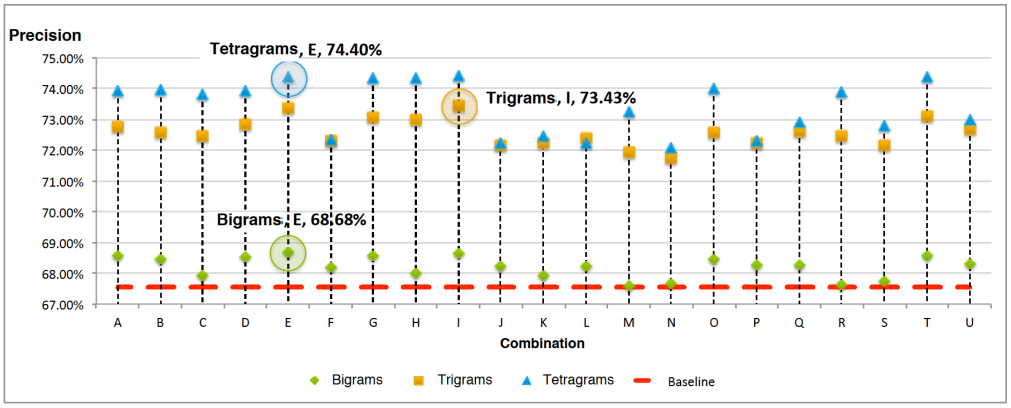

Fig. 10. Comparison of precision for syntactic dependency n-grams

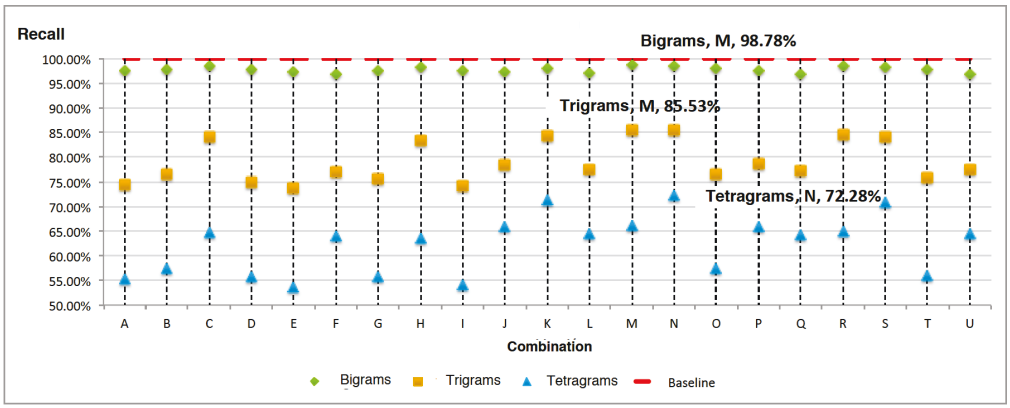

Fig. 11. Comparison of recall for syntactic dependency n-grams

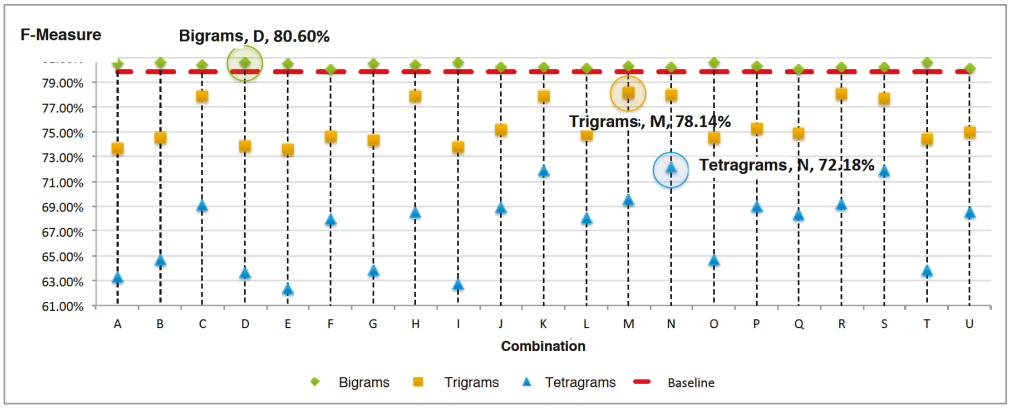

Fig. 12. Comparison of recall for syntactic dependency n-grams

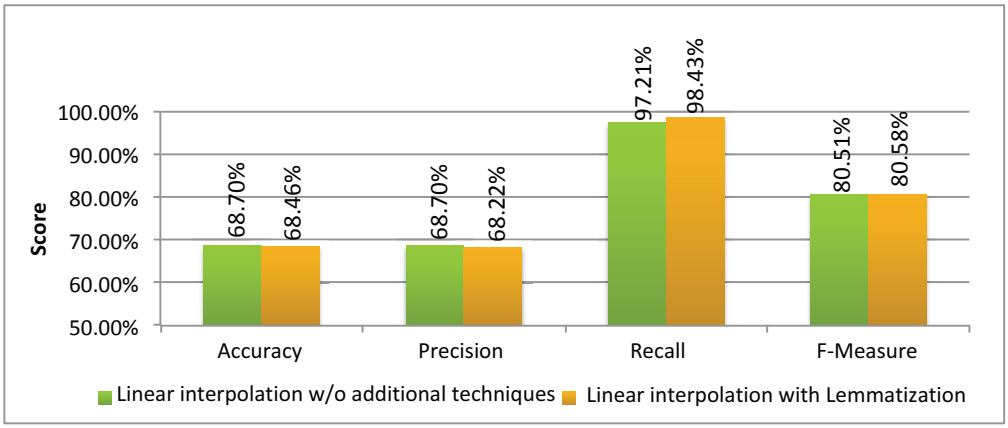

Fig. 13. Comparison of the linear combination method with and without stemming

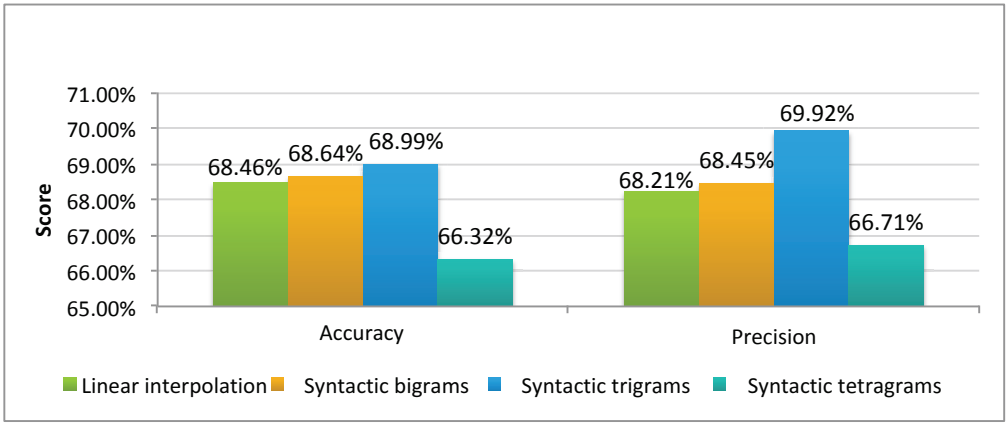

Fig. 14. Precision and Accuracy comparison for linear combination versus invididual n-grams

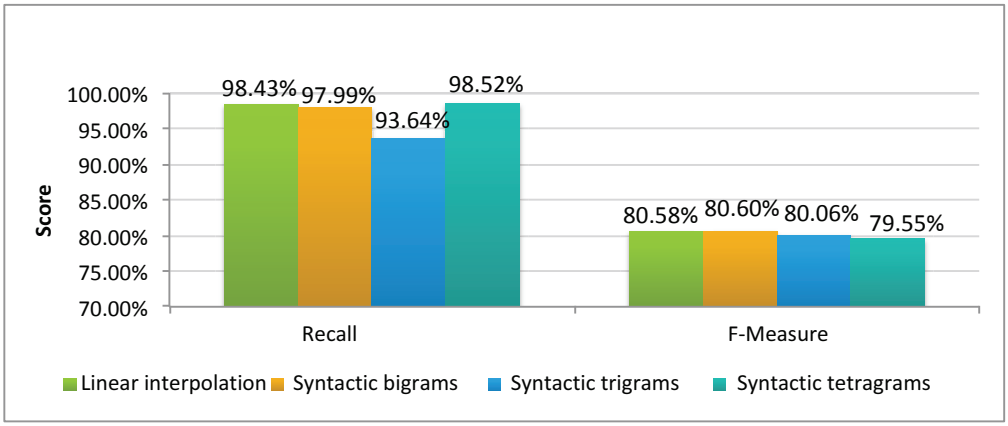

Fig. 15. Recall and F-measure comparison for linear combination versus invididual n-grams

Table 14. Best results for the similarity matrix method for the train and test sets

Corpus	Accuracy	Precision	Recall	F-measure	Thresnold
Train	68.25%	68.77%	97.05%	80.50%	0.20
Test	68.46%	68.62%	96.86%	80.33%	0.20

place, considering the F-measure, and in third place, considering recall. On the other hand, we can observe in Figure 17 that the similarity matrix method is able to obtain a better precision compared with simple syntactic dependency bigrams.

3.3 Syntactic Constituent N-grams

Similarly to syntactic dependency n-grams, we performed two different sets of experiments, the first one regarding the optimal thresholds per NLP technique combination, and then a fixed common threshold for all n-grams (0.55 in this case).

3.3.1 Syntactic Constituent Bigrams

In this section, we present our results for the training and test sets, shown in tables 15 and 16 respectively.

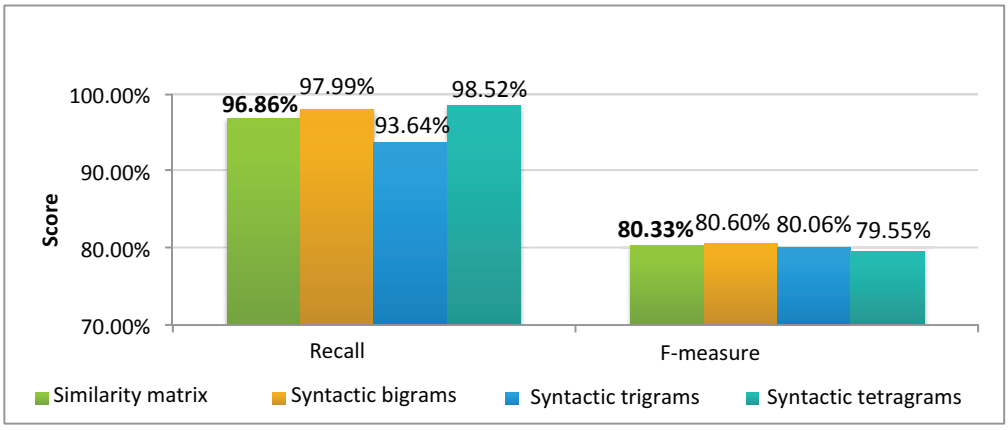

Fig. 16. Comparison of recall and F-measure for the similarity matrix method and simple syntactic dependency n-grams

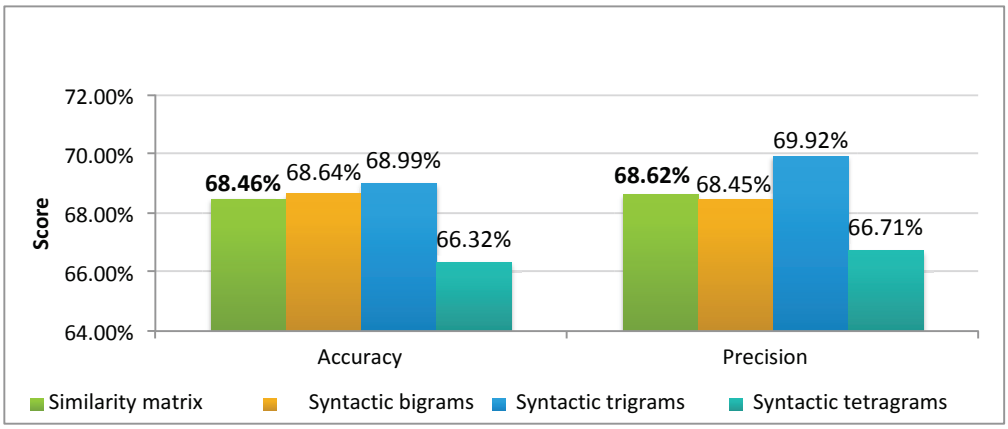

Fig. 17. Comparison of accuracy and precision for the similarity matrix method and simple syntactic dependency n-grams

We can see that the combination of NLP techniques of **stemming** and **Lin's similarity** ("M" key) helps to obtain the best score in the training set with a threshold of 0.45; on the other hand, by combining **stemming** and **synonyms** ("O" key) we get the best score for the test set (82.41%), using a threshold of 0.5. This is in fact the best score we found for paraphrase recognition with regard to any combination and proposed method described in this work.

3.3.2 Syntactic Constituent Trigrams

Tables 17 and 18 show our results for the train and test sets for syntactic constituent trigrams.

Based on the previous tables, we can see that using Lin's similarity measure ("C" key) helps to obtain the best F-measure with a threshold of 0.50 for the training set. In a similar way, combining Lin's similarity measure and negation ("H" key) produces the best F-score for the test set with a threshold of 0.50.

3.3.3 Syntactic Constituent Tetragrams

In tables 19 and 20 we present our results for syntactic constituent tetragrams on the training and test sets, respectively.

Table 15. Results for the training set for syntactic constituent bigrams

Key	Stemming	Lin's similarity	Synonyms	Negation	Stopwords removal	Accuracy	Precision	Recall	F-measure	Threshold
A						70.16%	69.91%	98.00%	81.60%	0.55
B	✓					70.73%	70.90%	96.11%	81.60%	0.50
C		✓				71.73%	72.30%	94.26%	81.83%	0.45
D			✓			70.73%	70.92%	96.04%	81.59%	0.50
E				✓		70.26%	70.18%	97.31%	81.55%	0.55
F					✓	69.06%	68.90%	98.76%	81.17%	0.60
G	✓			✓		70.82%	71.20%	95.38%	81.54%	0.50
H		✓		✓		71.76%	72.60%	93.46%	81.72%	0.45
I			✓	✓		70.14%	70.06%	97.42%	81.50%	0.55
J	✓				✓	69.03%	68.84%	98.91%	81.18%	0.60
K		✓			✓	69.94%	70.14%	96.62%	81.28%	0.50
L			✓		✓	69.08%	68.88%	98.91%	81.21%	0.60
M	✓	✓				**71.66%**	**71.93%**	**95.16%**	**81.93%**	**0.45**
N	✓	✓			✓	69.89%	69.95%	97.16%	81.34%	0.50
O	✓		✓			70.73%	70.86%	96.22%	81.62%	0.50
P	✓		✓		✓	70.36%	70.98%	94.91%	81.22%	0.50
Q	✓			✓	✓	70.68%	71.57%	93.86%	81.21%	0.50
R	✓	✓		✓		71.73%	72.27%	94.33%	81.84%	0.45
S	✓	✓		✓	✓	70.09%	70.32%	96.40%	81.32%	0.50
T	✓		✓	✓		70.82%	71.16%	95.49%	81.55%	0.50
U	✓		✓	✓	✓	70.75%	71.53%	94.18%	81.31%	0.50
Baseline						67.54%	67.54%	100.00%	80.62%	–

Considering the previously presented results, for the training set the highest F-measure can be obtained with a threshold of 0.90 ("D" key), while for the test set, the combination of synonyms and negation ("I" key) obtains the highest score with the same threshold.

3.3.4 Analysis and Evaluation for Syntactic Constituent N-grams

We used mainly the F-measure for comparison since it is used in most works for paraphrase recognition, and it allows a direct comparison with the same works that use the Microsoft Research Paraphrase Corpus. In Figure 18, we can see that all proposed combinations outperform baseline, being the best score **82.42**%. This measure itself was the highest score obtained among all our proposed methods and combination of techniques. This particular combination uses **stemming** and **synonyms** ("O" key) for the test set. Additionally, for the training set, **stemming** and **Lin's similarity** measure provide the best results, with an F-measure of 81.93%. Because of this, we can conclude that, in general, **stemming** is very important for paraphrase recognition using syntactic constituent n-grams.

The worst combination was stemming, negation and stopwords removal for the test set with

Table 16. Results for the test set for syntactic constituent bigrams

Key	Stemming	Lin's similarity	Synonyms	Negation	Stopwords removal	Accuracy	Precision	Recall	F-measure	Threshold
A						70.60%	69.77%	98.43%	81.66%	0.55
B	✓					72.17%	71.33%	97.21%	82.28%	0.50
C		✓				72.40%	72.38%	94.59%	82.01%	0.45
D			✓			72.34%	71.44%	97.29%	82.39%	0.50
E				✓		70.78%	70.13%	97.64%	81.63%	0.55
F					✓	68.23%	67.94%	98.86%	80.53%	0.60
G	✓			✓		72.23%	71.74%	96.07%	82.14%	0.50
H		✓		✓		72.34%	72.78%	93.28%	81.77%	0.45
I			✓	✓		70.78%	70.10%	97.73%	81.64%	0.55
J	✓				✓	68.17%	67.86%	99.04%	80.53%	0.60
K		✓			✓	69.73%	69.61%	96.68%	80.94%	0.50
L			✓		✓	68.34%	67.98%	99.04%	80.62%	0.60
M	✓	✓				71.82%	71.64%	95.37%	81.82%	0.45
N	✓	✓			✓	69.39%	69.18%	97.29%	80.86%	0.50
O	✓		✓			**72.34%**	**71.39%**	**97.47%**	**82.41%**	**0.50**
P	✓		✓		✓	70.20%	70.59%	94.59%	80.84%	0.50
Q	✓			✓	✓	70.02%	70.94%	93.02%	80.49%	0.50
R	✓	✓		✓		71.71%	72.01%	93.98%	81.54%	0.45
S	✓	✓		✓	✓	69.33%	69.55%	95.81%	80.60%	0.50
T	✓		✓	✓		72.40%	71.79%	96.33%	82.27%	0.50
U	✓		✓	✓	✓	70.02%	70.88%	93.19%	80.52%	0.50
Baseline						66.49%	66.49%	100.00%	79.87%	–

an F-measure of 80.50% ("Q" key). Regarding the syntactic constituent trigrams, we found that all combinations outperform baseline (see Figure 19), obtaining an F-measure of up to 81.91% by using Lin's similarity measure and negation ("H" key), improving even over the best combination of syntactic dependency analysis (80.60%). For the training set, by using Lin's similarity measure only ("C" key) we obtain the best score, being an F-measure of 81.48%. In general, we could say that the **Lin's similarity** measure is useful for syntactic constituent trigrams for paraphrase recognition. On the other hand, the worst performance was obtained by three different combinations: stemming and stopwords removal; synonyms and stopwords removal; and stemming, synonyms and stopwords removal. All of them yield an F-measure of 80.28%; nevertheless, such values are higher than baseline. Lastly, see Figure 20 for syntactic constituent tetragrams. We can see that only 8 out of 21 combinations outperform baseline. For the test set, the combination of **synonyms** and **negation** ("I" key) yields the best results, with an F-measure of **80.32%**. For the training set, using **synonyms** only ("D" key) yields the best results, with an F-measure of **80.74%**. This suggests that **synonyms** is a helpful technique for paraphrase recognition using syntactic constituent tetragrams.

Table 17. Results for syntactic constituent trigrams with the training set

Key	Stemming	Lin's similarity	Synonyms	Negation	Stopwords removal	Accuracy	Precision	Recall	F-measure	Threshold
A						70.21%	70.25%	96.94%	81.47%	0.55
B	✓					70.04%	70.06%	97.16%	81.41%	0.55
C		✓				**70.28%**	**70.35%**	**96.80%**	**81.48%**	**0.50**
D			✓			70.04%	70.09%	97.05%	81.40%	0.55
E				✓		70.28%	70.56%	96.11%	81.37%	0.55
F					✓	68.57%	68.46%	99.12%	80.99%	0.65
G	✓			✓		70.19%	70.38%	96.44%	81.37%	0.55
H		✓		✓		70.36%	70.67%	95.93%	81.38%	0.50
I			✓	✓		70.14%	70.40%	96.25%	81.32%	0.55
J	✓				✓	68.30%	68.11%	99.78%	80.96%	0.70
K		✓			✓	69.35%	69.69%	96.65%	80.99%	0.55
L			✓		✓	68.35%	68.13%	99.81%	80.99%	0.70
M	✓	✓				70.09%	70.04%	97.34%	81.47%	0.50
N	✓	✓			✓	69.28%	69.47%	97.23%	81.04%	0.55
O	✓		✓			70.06%	70.06%	97.23%	81.44%	0.55
P	✓		✓		✓	68.32%	68.12%	99.81%	80.97%	0.70
Q	✓			✓	✓	68.98%	69.26%	97.23%	80.90%	0.60
R	✓	✓		✓		70.21%	70.38%	96.51%	81.40%	0.50
S	✓	✓		✓	✓	69.40%	69.81%	96.36%	80.97%	0.55
T	✓		✓	✓		70.21%	70.38%	96.51%	81.40%	0.55
U	✓		✓	✓	✓	68.32%	68.22%	99.38%	80.91%	0.70
Baseline						67.54%	67.54%	100.00%	80.62%	–

Finally, we found 9 combinations that yield the same value for the F-measure, 79.83%, that was the lowest obtained score. All of those combinations use stopwords removal, along with the following techniques:

— Stopwords removal only ("F" key),
— Stemming ("J" key),
— Lin similarity "K" key),
— Synonyms ("L" key),
— Stemming, Lin similarity ("N" key),
— Stemming, synonyms ("P" key),
— Stemming, negation ("Q" key),
— Stemming, Lin's similarity, negation ("S" key),
— Stemming, synonyms, negation ("U" key).

Table 21 shows the best and worst combinations for the syntactic constituent n-grams in the test sets. Here we can see the interesting phenomenon that the same NLP technique can be part of the best and the worst combination at the same time, as described in Section 3.2.4.

In order to directly compare syntactic constituent n-grams, we use a fixed threshold of 0.55. This comparison, the same as in syntactic dependency n-grams, was done for syntactic constituent bigrams, trigrams, and tegragrams considering different combinations of NLP techniques.

Table 18. Results for syntactic constituent trigrams with the test set

Key	Stemming	Lin's similarity	Synonyms	Negation	Stopwords removal	Accuracy	Precision	Recall	F-measure	Threshold
A						70.43%	70.19%	96.51%	81.27%	0.55
B	✓					70.49%	70.18%	96.68%	81.33%	0.55
C		✓				71.24%	70.53%	97.47%	81.84%	0.50
D			✓			70.43%	70.17%	96.59%	81.29%	0.55
E				✓		70.78%	70.80%	95.37%	81.27%	0.55
F					✓	67.76%	67.55%	99.12%	80.35%	0.65
G	✓			✓		70.89%	70.81%	95.64%	81.37%	0.55
H		✓		✓		**71.71%**	**71.24%**	**96.33%**	**81.91%**	**0.50**
I			✓	✓		70.78%	70.78%	95.46%	81.29%	0.55
J	✓				✓	67.42%	67.17%	99.73%	80.28%	0.70
K		✓			✓	69.10%	69.09%	96.86%	80.65%	0.55
L			✓		✓	67.42%	67.17%	99.73%	80.28%	0.70
M	✓	✓				70.49%	69.91%	97.64%	81.48%	0.50
N	✓	✓			✓	69.21%	69.01%	97.47%	80.80%	0.55
O	✓		✓			70.49%	70.16%	96.77%	81.34%	0.55
P	✓		✓		✓	67.42%	67.17%	99.73%	80.28%	0.70
Q	✓			✓	✓	68.40%	68.51%	97.12%	80.34%	0.60
R	✓	✓		✓		70.89%	70.52%	96.59%	81.53%	0.50
S	✓	✓		✓	✓	69.39%	69.52%	96.07%	80.67%	0.55
T	✓		✓	✓		70.89%	70.79%	95.72%	81.39%	0.55
U	✓		✓	✓	✓	67.76%	67.49%	99.38%	80.39%	0.70
Baseline						66.49%	66.49%	100.00%	79.87%	–

First we will examine **accuracy**; see Figure 21. We can see in general a good performance of the syntactic constituent n-grams for the test set, outperforming most of them the baseline.

Bigrams and trigrams yield the best scores for this evaluation measure. Regarding **precision**, the syntactic constituent n-grams perform similarly to the syntactic dependency n-grams: tetragrams have better scores than bigrams and trigrams; see Figure 22.

The highest value of **recall**, is always obtained by the baseline. Figure 23 shows for recall a similar behavior to accuracy: bigrams and trigrams yield a better performance than tetragrams.

Finally for the **F-measure**, we have the same situation: bigrams and trigrams have a better performance than tetragrams; see Figure 24.

From these results, we conclude that syntactic constituent bigrams and trigrams have an advantage over tetragrams. In a similar way, syntactic constituent n-grams outperform syntactic dependency n-grams for this task.

The kind of syntactic constituent n-grams to be used depends, however, on the particular application to be developed. In general, taking into account the general score, we consider syntactic constituent trigrams as the most useful ones.

Table 19. Results for syntactic constituent tetragrams with the training set

Key	Stemming	Lin's similarity	Synonyms	Negation	Stopwords removal	Accuracy	Precision	Recall	F-measure	Threshold
A						67.93%	67.92%	99.52%	80.74%	0.90
B	✓					67.90%	67.90%	99.52%	80.73%	0.90
C		✓				67.54%	67.54%	100.00%	80.62%	0.95
D			✓			**67.93%**	**67.92%**	**99.52%**	**80.74%**	**0.90**
E				✓		67.88%	67.90%	99.45%	80.70%	0.90
F					✓	67.54%	67.54%	100.00%	80.62%	0.95
G	✓			✓		67.86%	67.88%	99.45%	80.69%	0.90
H		✓		✓		67.54%	67.54%	100.00%	80.62%	0.95
I			✓	✓		67.88%	67.90%	99.45%	80.70%	0.90
J	✓				✓	67.54%	67.54%	100.00%	80.62%	0.95
K		✓			✓	67.54%	67.54%	100.00%	80.62%	0.95
L			✓		✓	67.54%	67.54%	100.00%	80.62%	0.95
M	✓	✓				67.54%	67.54%	100.00%	80.62%	0.95
N	✓	✓			✓	67.54%	67.54%	100.00%	80.62%	0.95
O	✓		✓			67.90%	67.90%	99.52%	80.73%	0.90
P	✓		✓		✓	67.54%	67.54%	100.00%	80.62%	0.95
Q	✓			✓	✓	67.54%	67.54%	100.00%	80.62%	0.95
R	✓	✓		✓		67.54%	67.54%	100.00%	80.62%	0.95
S	✓	✓		✓	✓	67.54%	67.54%	100.00%	80.62%	0.95
T	✓		✓	✓		67.86%	67.88%	99.45%	80.69%	0.90
U	✓		✓	✓	✓	67.54%	67.54%	100.00%	80.62%	0.95
Baseline						67.54%	67.54%	100.00%	80.62%	–

3.4 Unigrams: A Special Case

Syntactic unigrams of words are a special case, since traditional n-grams, syntactic dependency unigrams and syntactic constituent n-grams are the same. For example, for the expression "*The Aztecs lived in Tenochtitlan*", traditional unigrams which build that sentence: *The*, *Aztecs*, *lived*, *in*, *Tenochtitlan*, are the same than the syntactic dependency and constituent unigrams; see Figure 25.

We compared unigrams (traditional or syntactic), without additional NLP-techniques, with the best results of syntactic dependency and constituent bigrams. Figure 26 shows results for all four evaluation measures. We found a better performance of unigrams in three out of four measures. This is probably due to the fact that the Microsoft Research Paraphrase Corpus has a great quantity of lexical overlap, and less quantity of syntactic diversity as it is mentioned by [29] and [28].

3.5 Syntactic Dependency N-grams vs. Syntactic Constituent N-grams

Here we present a comparison of the two kinds of syntactic n-grams studied in this work: syntactic dependency n-grams and syntactic constituent n-grams. This comparison was done for bigrams,

Table 20. Results for synactic constituent tetragrams with the test set

Key	Stemming	Lin's similarity	Synonyms	Negation	Stopwords removal	Accuracy	Precision	Recall	F-measure	Threshold
A						67.18%	67.09%	99.38%	80.11%	0.90
B	✓					67.18%	67.09%	99.38%	80.11%	0.90
C		✓				66.49%	66.49%	100.00%	79.87%	0.95
D			✓			67.24%	67.11%	99.47%	80.15%	0.90
E				✓		67.53%	67.33%	99.38%	80.28%	0.90
F					✓	66.43%	66.47%	99.91%	79.83%	0.95
G	✓			✓		67.53%	67.33%	99.38%	80.28%	0.90
H		✓		✓		66.49%	66.49%	100.00%	79.87%	0.95
I			✓	✓		**67.59%**	**67.35%**	**99.47%**	**80.32%**	**0.90**
J	✓				✓	66.43%	66.47%	99.91%	79.83%	0.95
K		✓			✓	66.43%	66.47%	99.91%	79.83%	0.95
L			✓		✓	66.43%	66.47%	99.91%	79.83%	0.95
M	✓	✓				66.49%	66.49%	100.00%	79.87%	0.95
N	✓	✓			✓	66.43%	66.47%	99.91%	79.83%	0.95
O	✓		✓			67.24%	67.11%	99.47%	80.15%	0.90
P	✓		✓		✓	66.43%	66.47%	99.91%	79.83%	0.95
Q	✓			✓	✓	66.43%	66.47%	99.91%	79.83%	0.95
R	✓	✓		✓		66.49%	66.49%	100.00%	79.87%	0.95
S	✓	✓		✓	✓	66.43%	66.47%	99.91%	79.83%	0.95
T	✓		✓	✓		67.59%	67.35%	99.47%	80.32%	0.90
U	✓		✓	✓	✓	66.43%	66.47%	99.91%	79.83%	0.95
Baseline						66.49%	66.49%	100.00%	79.87%	–

trigrams and tetragrams, considering the combinations of NLP techniques where the best results were obtained in the test corpus; such results can be found in tables from sections 3.2 and 3.3.)

Regarding syntactic bigrams, Figure 27 shows that syntactic constituent bigrams have better performance in 3 out of 4 evaluation measures, except for recall. However, this latter is only slightly improved with respect to other measures. In general, syntactic constituent bigrams present a remarkable difference.

For syntactic trigrams, constituents obtain also the best results, outperforming syntactic dependency trigrams in all 4 evaluation measures, as shown in Figure 28.

On the other hand, we can see also in Figure 28 that, despite constituent analysis obtains better scores in the evaluation measures, the differences are not as great as in the case of syntactic bigrams.

Finally, for syntactic tetragrams we have the same situation that for trigrams: syntactic constituent tetragrams have a better performance than syntactic dependency tetragrams in all 4 evaluation measures. The difference is even smaller between them, compared with the difference for syntactic bigrams and trigrams; see Figure 29.

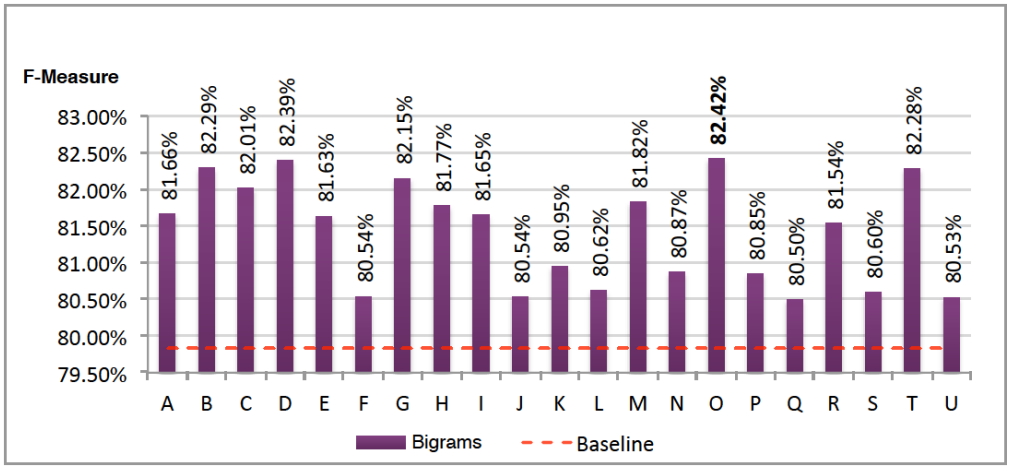

Fig. 18. Comparison for syntactic constituent bigrams with the test set

Table 21. Summary of the best and the worst techniques for syntactic constituent n-grams

	Best combination	**Worst combination**
Syntactic bigrams	Stemming + synonyms	Stemming + negation + stopwords removal
Syntactic trigrams	Lin similarity + negation	-Stemming + stopwords removal; -Synonyms + stopwords removal; -Stemming + synonyms + stop words removal
Syntactic tetragrams	Synonyms + negation	All combinations that include stopwords removal

In conclusion, we found a better performance with syntactic constituent n-grams compared with syntactic dependency n-grams for paraphrase recognition. It is important to mention, however, that it might be a different kind of syntactic dependency n-gram not explored in this work (*cf.* [19]), that could achieve better results than constituents.

3.6 Results of Syntactic N-grams vs. Traditional N-grams

In this section, we present a comparison between traditional n-grams and syntactic dependency and constituent n-grams. This comparison is done without NLP auxiliary techniques, using the optimal threshold for each one. Our goal for this comparison is to dtermine if syntactic n-grams are helpful or not for the task of paraphrase recognition.

Figure 30 shows results for traditional bigrams, syntactic dependency and constituent bigrams. We can see syntactic n-grams have better results, being dependency or constituent-based ones.

An important aspect to notice is that syntactic constituent bigrams easily outperform traditional bigrams in all 4 evaluation measures; however, traditional bigrams outperform syntactic dependency bigrams in recall.

Table 22 shows hits (true positives and negatives) and misses (false positives and negatives) for each kind of bigram. We can see that the advantage of syntactic bigrams lies in the decrease of false positives and false negatives, probably due to the requirement of more exact matches where dependency relationships and part of speech tags are included.

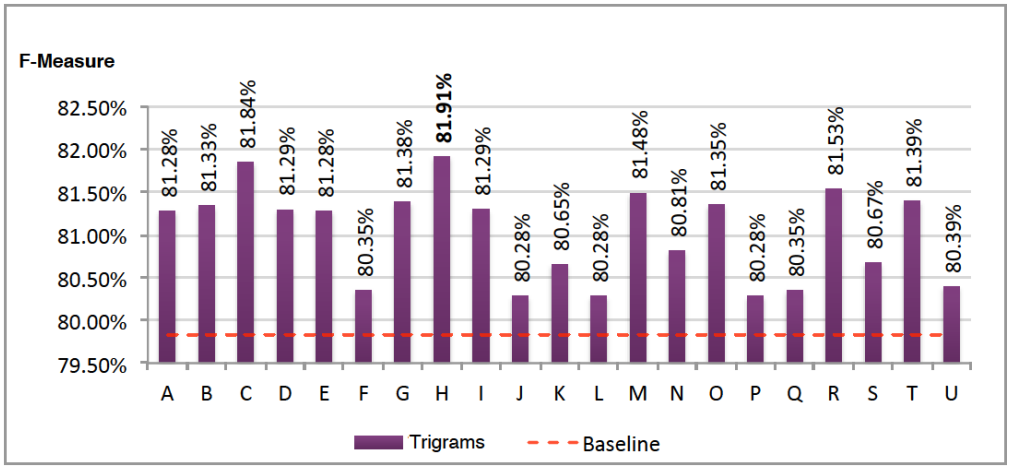

Fig. 19. Comparison for syntactic constituent bigrams techniques with the test set

Table 22. Hits and misses of traditional and syntactic bigrams

Kind	True positives	True negatives	False positives	False negatives
Traditional bigrams	1120	44	534	27
Syntactic dependency bigrams	1119	65	513	28
Syntactic constituent bigrams	1129	89	489	18

For the case of trigrams, it is not possible to say there is an absolute winner, because on one hand, traditional trigrams outperform syntactic dependency trigrams in F-measure; however, this is due to the lower recall of syntactic dependency trigrams. This can be seen in Figure 31.

Nevertheless, syntactic constituent trigrams outperform traditional trigrams in 3 out of 4 evaluation measures (that is, other measures than recall). Table 23 shows hits and misses for the three compared models. Syntactic n-grams are able to identify more true negatives and less false positives. The higher scores of traditional n-grams are consequence of the unbalanced corpus, because most of the input pairs are marked as a true paraphrase—this is seen as a near-100% recall), while syntactic n-grams are less biased.

Finally, for tetragrams he have a similar behavior to trigrams. Traditional tetragrams outperform syntactic dependency tetragrams, but tratitional tetragrams are outperformed in turn by the syntactic constituent tetragrams in 3 out of 4 evaluation measures (except for recall, where both have the same performance). This can be seen in Figure 32.

Table 24 shows hits and misses for tetragrams. In this case, traditional tetragrams and syntactic constituents tetragrams are biased by the unbalanced corpus. Despite the syntactic constituent tetragrams yield the best results, the performance of syntactic dependency tetragrams should not be ignored, since it has the best performance for negative pairs.

According to the previously presented results, we can conclude that syntactic n-grams are more useful than traditional n-grams for paraphrase recognition. Particularly, syntactic constituent n-grams have a clearer benefit in all n-gram sizes studied.

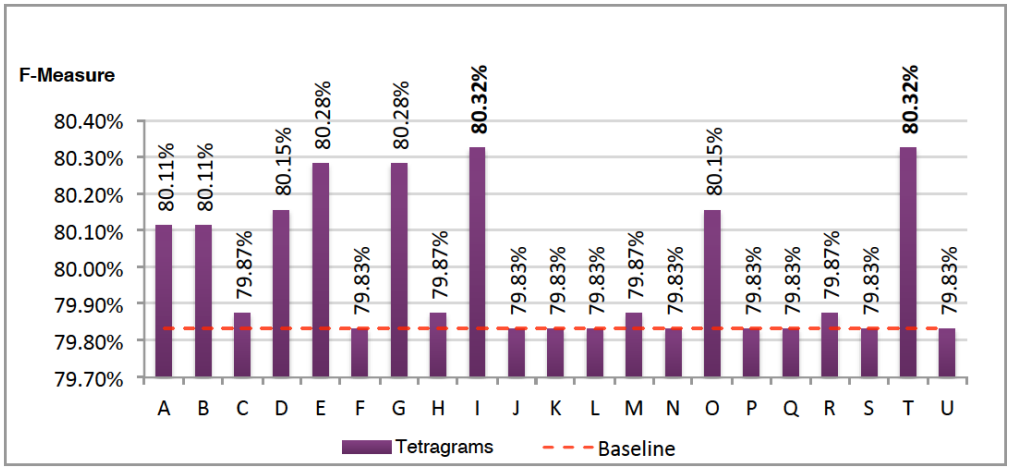

Fig. 20. Comparison of syntactic constituent tetragrams evaluated with the test set

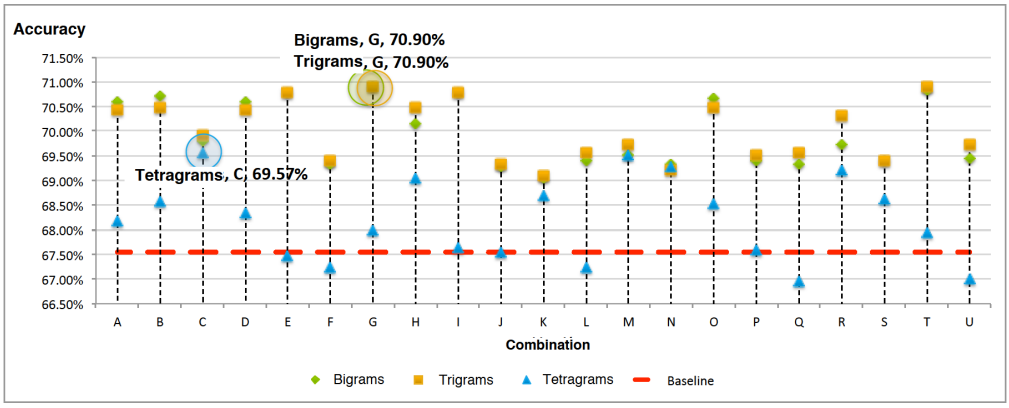

Fig. 21. Comparison of accuracy for syntactic constituent n-grams

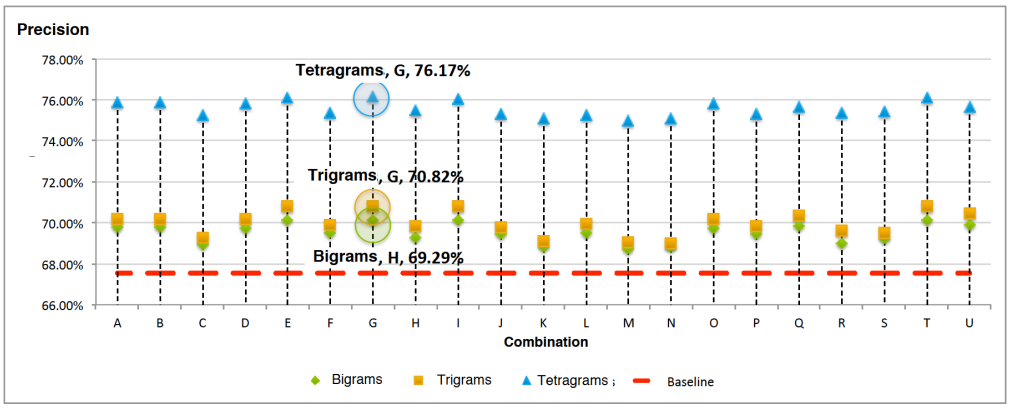

Fig. 22. Comparison of precision of syntactic constituent n-grams

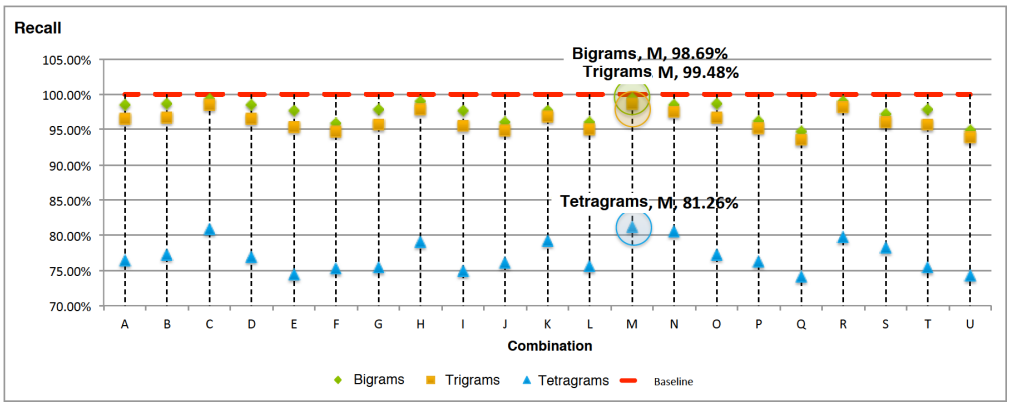

Fig. 23. Comparison of recall for syntactic constituent n-grams

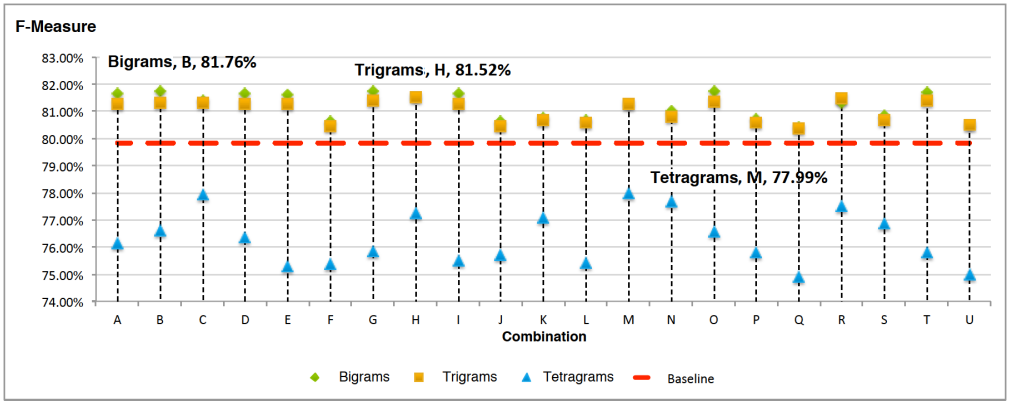

Fig. 24. Comparison of F-measure for syntactic constituent n-grams

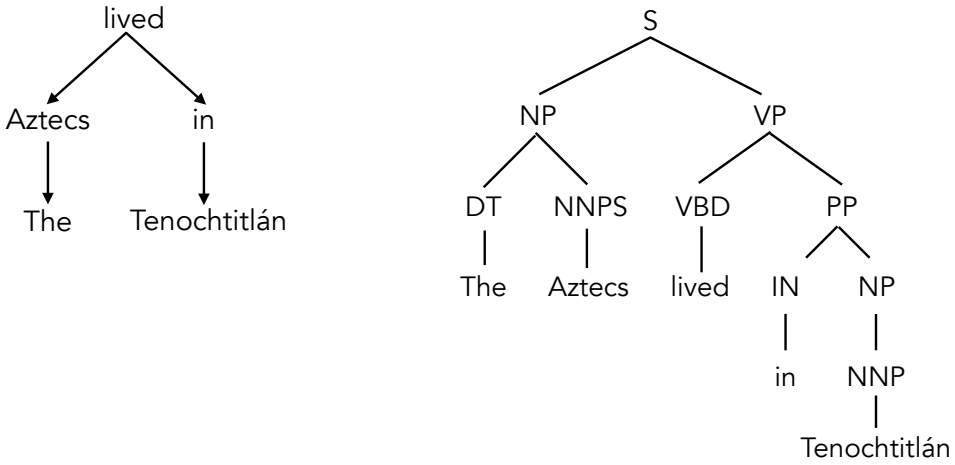

Fig. 25. Syntactic unigrams

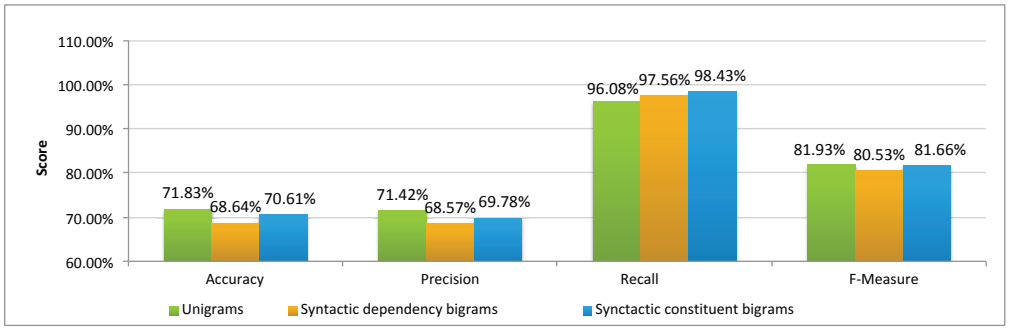

Fig. 26. Comparison of unigrams and syntactic bigrams

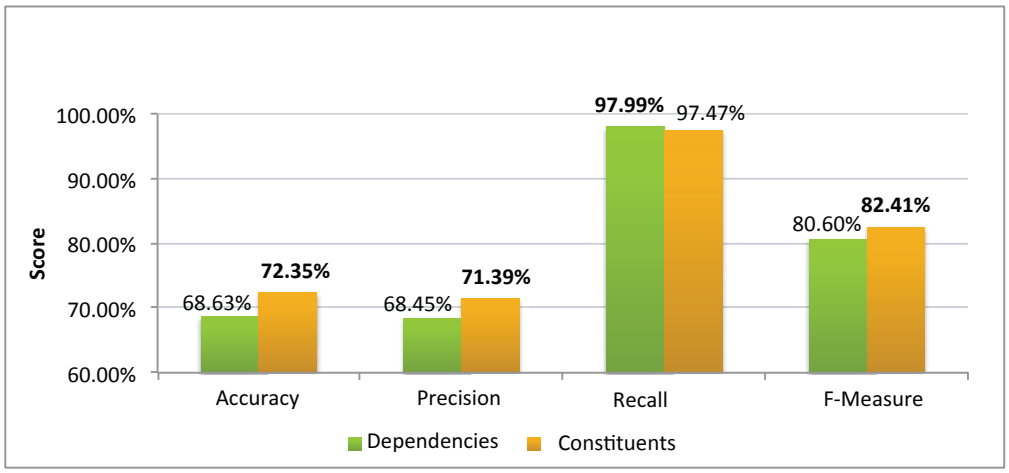

Fig. 27. Comparison of syntactic bigrams

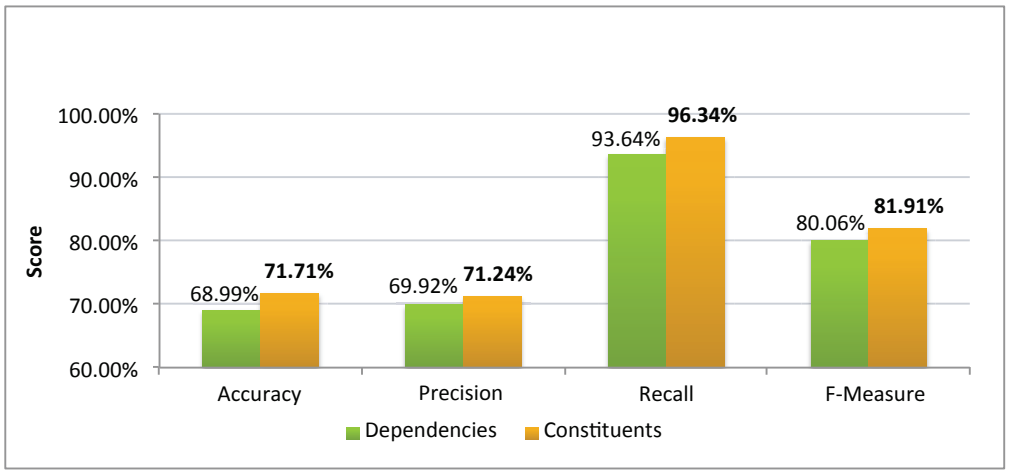

Fig. 28. Comparison of syntactic trigrams

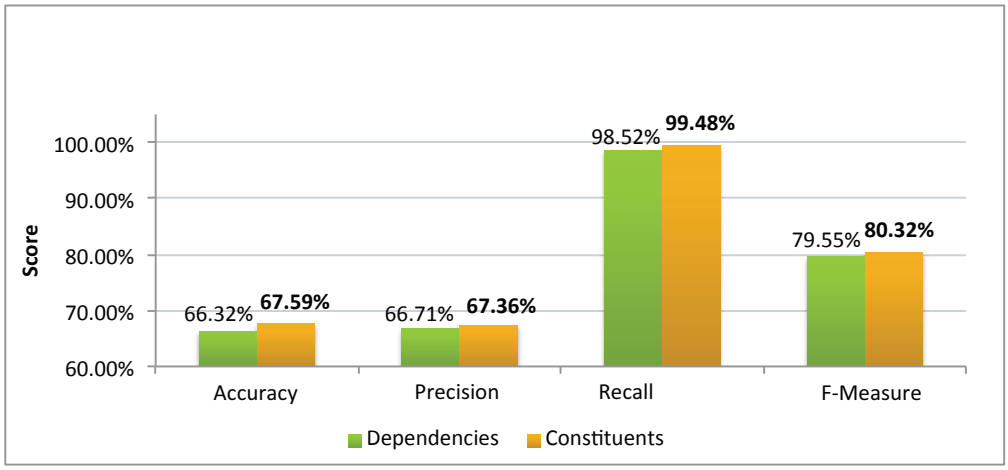

Fig. 29. Comparison of syntactic tetragrams

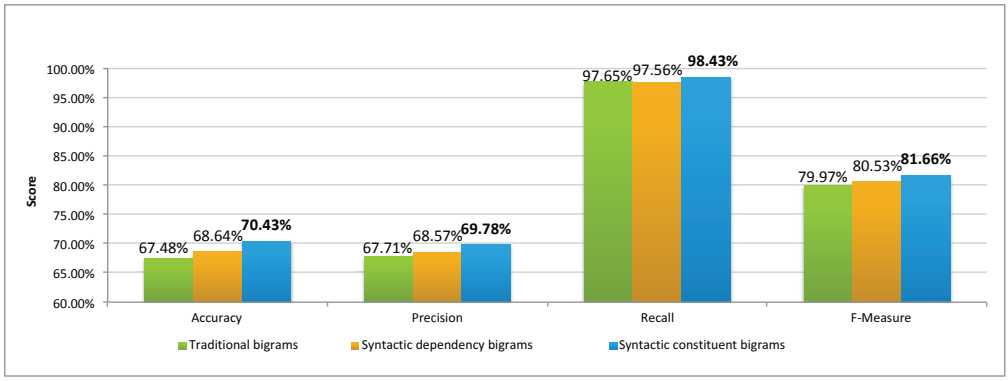

Fig. 30. Comparison of traditional and syntactic bigrams

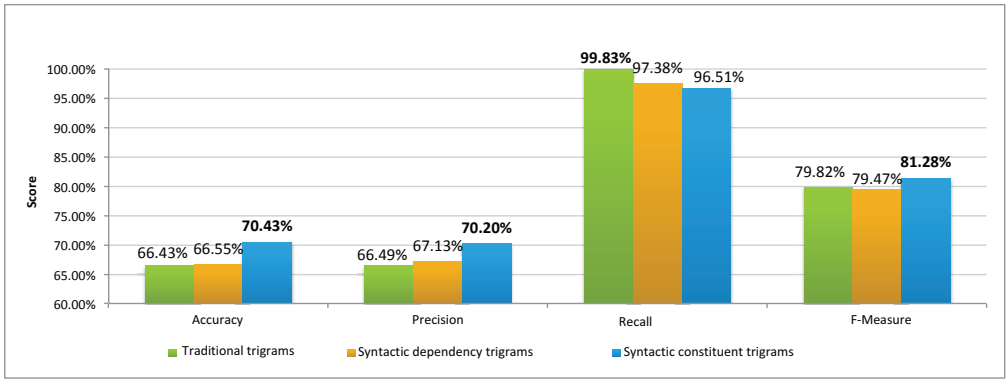

Fig. 31. Comparison of traditional and syntactic trigrams

Table 23. Hits and misses of traditional and syntactic trigrams

Kind	True positives	True negatives	False positives	False negatives
traditional trigrams	1145	1	577	2
Syntactic dependency trigrams	1117	31	547	30
Constituent syntactic trigrams	1107	108	470	40

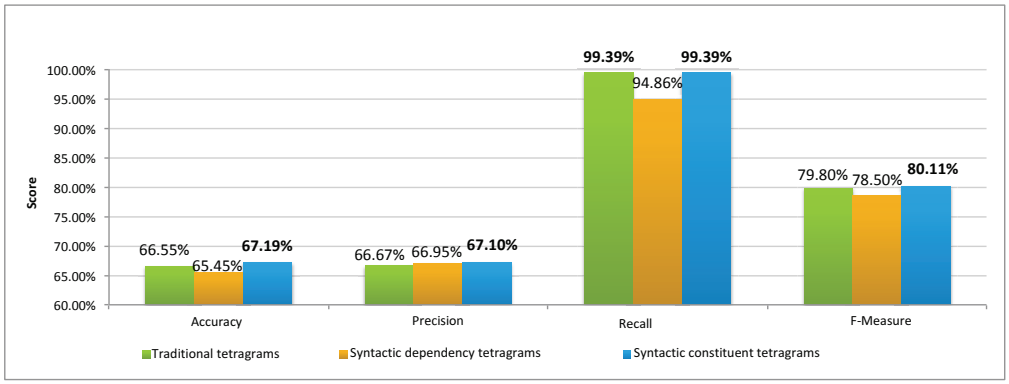

Fig. 32. Comparison of traditional and syntactic trigrams

Table 24. Hits and misses of traditional and syntactic trigrams

Kind	True positives	True negatives	False positives	False negatives
Traditional tetragrams	1140	8	570	7
Syntactic dependency tetragrams	1088	41	537	59
Syntactic constituent tetragrams	1140	19	559	7

4 Error Analysis

In this section, we analyze the most common error sources of the methods presented in this work. For this, we selected a random sample of 10 expression pairs that could not be correctly solved by any of the presented paraphrase recognition methods.

We have already mentioned several problems of unbalancedness of the Microsoft Research Paraphrase Corpus; however, during error analysis, we found another problem, which is the inconsistency in the classification of expression pairs; that is, some pairs are marked as TRUE in the corpus (train and test sections), even when additional information is present among them. On the other hand, some other pairs are tagged as FALSE because of the same reason, creating confusing features when classification is done. This might occur because the classification of each pair was left under consideration of each evaluator, as described in the paper about the creation of this corpus [3].

Let us present some examples of this aforementioned problem. Two pair of expressions are shown, where the first pair has been tagged as a true paraphrase, while the second one has been tagged as false:

Pair 1: tagged as true.

Expression 1: *Google's investors include prominent VC firms Kleiner Perkins Caufield & Byers and Sequoia Capital, the paper noted.*

Expression 2: *Google's early stage backers in include California-based Stanford University and VC firms Kleiner Perkins and Sequoia Capital.*

Pair 2: tagged as false.

Expression 1: *Reuters witnesses said many houses had been flattened and the city squares were packed with crying children and the homeless, huddled in blankets to protect them from the cold.*

Expression 2: *Reuters witnesses said public squares were packed with crying children and people left homeless, huddled in blankets to protect them from the cold.*

We have underlined the additional information present in the pair of expressions. We can see that this information is similar, but they have been tagged differently. This problem is partly handled by the difference threshold implemented by each paraphrase method presented in this work, however, additional problems were found. For the sake of clarity, the examples are shown at the lexical level, despite these phenomena are present at the syntactic level as well.

1. **Deceitful lexical and syntactic overlapping:** this error refers to expressions where the amount of lexical and syntactic overlapping is high, but the conveyed semantic information is not the same, causing the paraphrase pairs to be classified as true. For example:
   - **Expression 1:** *"Amnesty International has said that over the past 20 years it has collected information about 17,000 disappearances in Iraq but the actual figure may be much higher."*
   - **Expression 2:** *Amnesty International said that over the past 20 years it had collected information about 17,000 disappearances in Iraq.*

   We can see that the amount of lexical and syntactic overlap is high (underlined text). Despite of this, this pair was manually classified as false due to the semantic difference between the two expressions, because in the first text, the disappeared number of persons in Iraq could be more than 17,000, which is not mentioned in the second text.
2. **Lack of anaphora resolution:** Possible paraphrases were found, where one of the expressions contained an anaphora, while the other did not. This caused that false pairs were classified as true, due to the high level of syntactic and lexical overlap. For example:
   - **Expression 1:** *"This is America, my friends, and it should not happen here," he said to loud applause.*
   - **Expression 2:** *"This is America, my friends, and it should not happen here."*

   Additionaly, for this kind of pairs we found some inconsistency in the corpus. For example, the following expression was tagged as true (compare with the previous example, tagged as false):
   - **Expression 1:** *These are real crimes that hurt a lot of people.*
   - **Expression 2:** *"These are real crimes that disrupt the lives of real people," Smith said.*
3. **Temporal and spatial expressions:** Some of the sentences present some semantic similarity because they speak about similar subjects; however, they are different in place and time. For example:
   - **Expresión 1:** *Russ Britt is the Los Angeles Bureau Chief for CBS.MarketWatch.com.*
   - **Expresión 2:** *Emily Church is London bureau chief of CBS.MarketWatch.com.*

Both expressions talk about CBS.MarketWatch.com's bureau chief, but the spatial location of both expressions is different. This pair was marked as true by all our presented paraphrase methods due to the high syntactic overlap between the two expressions. However, if location was considered as a key difference feature between the two expressions, the pair would have been correctly classified as false.

4. **Numeric differences:** refers to the pairs of expressions where there is a numeric difference that was considered as similar, which causes the incorrect pair classification if they are marked as true parapras pairs. For example:

   – **Expression 1:** *The benchmark 10-year note US10YT=RR lost 11/32 in price, taking its yield to 3.21 percent from 3.17 percent late on Monday.*

   – **Expression 2:** *Further out the curve, the benchmark 10-year note US10YT=RR shed 18/32 in price, taking its yield to 3.24 percent from 3.17 percent.*

   Of course quantities were considered as different, but a greater penalization would have been useful, due to the semantic impact that this issue represents for this pair and similar ones, and thus, yielding a correct classification.

An important point is that, in a single pair, several causes of error can be mixed. Moreover, some additional causes pertaining to each analysis were found: for example, for dependency analysis, the main cause of error was the assignment of the root for each expression in a paraphrase pair, causing multiple differences in the syntactic n-grams, and finally yielding an incorrect classification of pairs being marked as false. An example follows.

Expression 1: *Monday's attacks Monday were among the deadliest against Americans since Sept. 11, 2001.*

Expression 2: *They were the deadliest terrorist attacks against Americans since September 11.*

In Table 25 we can see that the syntactic dependency n-grams are different because the root verb is different, causing the pair to be misclassified as false, when it is actually true.

In the same way, in the syntactic constituent analysis we found two kinds of recurrent error. The first one lies in assigning a different part of speech tag to a word and one of its derivation. For example *slim* and *slimness* correspond to an adjective and a noun respectively; however, these words are actually related between them. When they are marked with different tags, it is nos possible to relate them anymore, generating in turn incorrect classifications when a true pair is marked as false. On the other hand, the second problem is presented when for a single word, the Stanford Parser assigns different tags in each sentence, which is sometimes correct, but when it is incorrectly done, it derives in the same aforementioned problem, causing true pairs to be misclassified as false. For example:

Expression 1: *Seven of the nine major Democratic[JJ] presidential[JJ] candidates will address the forum.*

Expression 2: *Seven of nine Democratic[NN] candidates for president[NN] also said they would participate in the conference Monday.*

The first kind of problem is exemplified with the words *presidential* and *president*, where the syntactic role assigned to each one is different (the first one as adjective (JJ) and the second one as a noun (NN)). However, these words should present some semantic similarity. On the other hand, the word *Democratic* has a different tag in each expression, despite the syntactic role in both contexts is adjective. With this information, the syntactic constituent analysis ignores this coincidence. Table 26 shows the syntactic constituent n-grams for this example, where we can see that the pair would have been correctly classified as true if the highlighted syntactic n-grams would had been considered as equivalents.

Table 25. Syntactic dependency n-grams for the example expressions

Syntactic n-grams for expression 1	Syntactic n-grams for expression 2
attacks:monday	attacks:they
were:attacks	attacks:were
were:monday	attacks:the
root:were	attacks:deadliest
deadliest:the	attacks:terrorist
were:deadliest	**root:attacks**
deadliest:americans	attacks:americans
were:sept.	attacks:september
sept.:11	september:11
sept.:2001	

Table 26. Syntactic dependency n-grams for the example expressions

Syntactic n-grams for expression 1	Syntactic n-grams for expression 2
CD:seven	CD:seven
IN:of	IN:of
DT:the	CD:nine
CD:nine	**JJ:democratic**
JJ:major	NN:candidates
NN:democratic	IN:for
NN:presidential	**NN:president**
NN:candidates	RB:also
MD:will	VB:said
VB:address	PRP:they
DT:the	MD:would
NN:forum	VB:participate
	IN:in
	DT:the
	NN:conference
	NN:monday

Some of the causes of error presented in this section require a depper analysis for handling them. Even more, some of them may need the creation of a corpus without ambiguity in the tags assigned to the paraphrase pairs, so that clear criteria can be established for a correct classification.

5 Comparison with Other Works

First we present a brief summary of the best results of our proposed methods, and thereafter we present our position in the state of the art found for the paraphrase recognition task. Table 27 summarizes the best score for syntactic dependency and constituent bigrams, trigrams and tetragrams for the test set, as well as the best scores for the linear combination and similarity matrix methods, showing the best combination of NLP techniques as well.

According to the previous table, we can see that the best scores for syntactic dependency and constituent n-grams were found with syntactic bigrams.

Lastly, in tables 28 and 29 we present the final position reached by the best scores of syntactic

Table 27. Summary of the best scores for paraphrase recognition

Analysis	Recognition	Best combination	Score
Dependency	Syntactic bigrams	Sinónimos	80.60%
	Syntactic trigrams	Stemming, Lin, negation and stopwords removal	80.05%
	Syntactic tetragrams	Lin and stopwords removal	79.54%
	Linear combination	Stemming	80.58%
	Similarity matrix	basic (no additional NLP techniques)	80.33%
Constituents	Syntactic bigrams	Stemming and synonyms	82.41%
	Syntactic trigrams	Lin and negation	81.91%
	Syntactic tetragrams	Synonyms and negation	80.32%

dependency and constituents, with regard to supervised and unsupervised systems, respectively. As we are not using a particular machine learning method, we could say that our method is unsupervised; however, the syntactic parser uses hand-tagged data, so this part would be supervised. That is why we are not able to state that our method is fully unsupervised.

In tables 28 and 29 we can see that both kinds of syntactic analysis have a competitive performance for the paraphrase recognition task. Syntactic constituent n-grams stand out, obtaining globally the second and sixth position when compared with unsupervised and supervised methods respectively.

6 Conclusions and Future Work

Syntactic n-grams have shown to be an useful technique for paraphrase recognition. Our proposed methods outperformed several works in the state of the art. According to the way they were used, syntactic n-grams achieve a better performance compared with traditional n-grams in paraphrase recognition. Syntactic constituent n-grams yielded better scores than syntactic dependency n-grams, but we cannot conclude they are superior in all cases, because syntactic dependency n-grams could be implemented in different ways, or with other combination of NLP techniques not approached in this work so that they can improve their performance.

The best score was obtained with syntactic constituent bigrams, and the best combination of NLP techniques was synonyms and stemming. This confirms the hypothesis that synonyms clearly help paraphrase recognition. Sometimes a paraphrase can be built only by changing some words by their respective synonyms.

On the other hand, the worst score was found with syntactic dependency tetragrams and the NLP technique of negation. In many cases, our implementation of negation was not useful for this task. We leave as a future work to investigate deeper negation analysis techniques that could further improve results.

One of the difficulties we found in this work was the evaluation criteria for pairs of expressions in the Microsoft Research Paraphrase Corpus, because they were sometimes inconsistent, generating confusion.

As a future work we propose to use or build a new corpus as a reference for paraphrase recognition due to the disadvantages present in the current Microsoft Research's corpus, such as unbalancedness and uneven criteria. Additionally, we could explore with different syntactic analyzers aiming for a better performance; we plan to improve the similarity matrix method (for example by using a soft cosine Measure [21]) as well by exploring different ways to explore the similarity between a pair of syntactic bigrams.

Acknowledgements

Work done under support of CONACyT-SNI, SIP-IPN, COFAA-IPN, and PIFI-IPN.

Table 28. F-measure-based position of our proposal compared with supervised methods

Author	Accuracy	F-Measure	Learning method
[12]	77.4%	84.1%	Supervised
[24]	76.8%	83.6%	
[27]	75.6%	83.0%	
[2]	76.1%	82.7%	
[5]	75.0%	82.7%	
Constituents (2014)	**72.34%**	**82.41%**	
[11]	74.1%	82.3%	
[1]	73.0%	82.3%	
[26]	74.7%	81.8%	
[15]	72.0%	81.6%	
[6]	73.2%	81.3%	
Dependencies (2014)	**68.63%**	**80.60%**	
[8]	76.6%	79.6%	

Table 29. F-measure-based position of our proposal compared with unsupervised methods

Author	Accuracy	F-Measure	Learning method
[13]	76.17%	82.88%	Unsupervised
Constituents (2014)	**72.34%**	**82.41%**	
[4]	74.1%	82.4%	
[7]	72.6%	81.3%	
[14]	70.3%	81.3%	
Dependencies (2014)	**68.63%**	**80.60%**	
[16]	70.6%	80.5%	
[14]	65.4%	75.3%	

References

1. **Blacoe, W. & Lapata, M.** (**2012**). A Comparison of Vector-based Representations for Semantic Composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12. ACL, Stroudsburg, PA, USA, 546–556.
2. **Das, D. & Smith, N. A.** (**2009**). Paraphrase Identification As Probabilistic Quasi-synchronous Recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, volume 1 of *ACL '09*. ACL, Stroudsburg, PA, USA. ISBN 978-1-932432-45-9, 468–476.
3. **Dolan, B., Quirk, C., & Brockett, C.** (**2004**). Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04. ACL, Stroudsburg, PA, USA.
4. **Fernando, S. & Stevenson, M.** (**2008**). A Semantic Similarity Approach to Paraphrase Detection. In *Proceedings of the Computational Linguistics UK (CLUK 2008) 11th Annual Research Colloquium*.
5. **Finch, A., Hwang, Y., & Sumita, E.** (**2005**). Using Machine Translation Evaluation Techniques to Determine Sentence-level Semantic Equivalence. In *Proceedings of the Third International Workshop on Paraphrasing (IWP 2005)*. Jeju Island, South Korea, 17–24.
6. **Heilman, M. & Smith, N. A.** (**2010**). Tree Edit Models for Recognizing Textual Entailments, Paraphrases, and Answers to Questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL*, HLT '10.

ACL, Stroudsburg, PA, USA. ISBN 1-932432-65-5, 1011–1019.

7. **Islam, A. & Inkpen, D.** (**2007**). Semantic Similarity of Short Texts. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2007)*. John Benjamins Publishing, Borovets, Bulgaria, 291–297.
8. **Kozareva, Z. & Montoyo, A.** (**2006**). Paraphrase Identification on the Basis of Supervised Machine Learning Techniques. In *Advances in Natural Language Processing, 5th International Conference on NLP*, FinTAL '06. Springer-Verlag, Berlin, Heidelberg. ISBN 3-540-37334-9, 978-3-540-37334-6, 524–533.
9. **Lin, D.** (**1998**). An Information-Theoretic Definition of Similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN 1-55860-556-8, 296–304.
10. **Lin, D.** (**1998**). Dependency-based Evaluation of MINIPAR. In *Proceedings of the Workshop on the Evaluation of Parsing Systems*. Springer Netherlands, Granada, Spain.
11. **Lintean, M. & Rus, V.** (**2011**). Dissimilarity Kernels for Paraphrase Identification. In *Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference*. 263–268.
12. **Madnani, N., Tetreault, J., & Chodorow, M.** (**2012**). Re-examining Machine Translation Metrics for Paraphrase Identification. In *Proceedings of the 2012 Conference of the North American Chapter of the ACL: Human Language Technologies*, NAACL HLT '12. ACL, Stroudsburg, PA, USA. ISBN 978-1-937284-20-6, 182–190.
13. **Malakasiotis, P.** (**2009**). Paraphrase Recognition Using Machine Learning to Combine Similarity Measures. In *Proceedings of the ACL-IJCNLP 2009 Student Research Workshop*, ACLstudent '09. ACL, Stroudsburg, PA, USA, 27–35.
14. **Mihalcea, R., Corley, C., & Strapparava, C.** (**2006**). Corpus-based and Knowledge-based Measures of Text Semantic Similarity. In *Proceedings of the 21st National Conference on Artificial Intelligence*, volume 1 of *AAAI'06*. AAAI Press. ISBN 978-1-57735-281-5, 775–780.
15. **Qiu, L., Kan, M.-Y., & Chua, T.-S.** (**2006**). Paraphrase Recognition via Dissimilarity Significance Classification. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06. ACL, Stroudsburg, PA, USA. ISBN 1-932432-73-6, 18–26.
16. **Rus, V., McCarthy, P. M., & Lintean, M. C.** (**2008**). Paraphrase Identification with Lexico-Syntactic Graph Subsumption. In *Proceedings of the 21st International FLAIRS Conference*. Coconut Grove, FL., 201–206.
17. **Segura-Olivares, A., García, A., & Calvo, H.** (**2013**). Feature Analysis for Paraphrase Recognition and Textual Entailment. In *Research in Computing Science*, volume 70 of *Advances in Computational Linguistics (CICLing-2013)*. Kathmandu, Nepal, 119–144.
18. **Sidorov, G.** (**2013**). *Construcción no lineal de n-gramas en la lingüística computacional: n-gramas sintácticos, filtrados y generalizados*. Sociedad Mexicana de Inteligencia Artificial. ISBN 978-607-95367-9-4.
19. **Sidorov, G.** (**2013**). Non-continuous Syntactic N-grams. *Polibits*, 48, 67–75.
20. **Sidorov, G.** (**2013**). Syntactic Dependency Based N-grams in Rule Based Automatic English as Second Language Grammar Correction. *International Journal of Computational Linguistics and Applications*, 4(2), 169–188.
21. **Sidorov, G., Gelbukh, A., Gómez-Adorno, H., & Pinto, D.** (**2014**). Soft Similarity and Soft Cosine Measure: Similarity of Features in Vector Space Model. *Computación y Sistemas*, 18(3).
22. **Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A., & Chanona-Hernández, L.** (**2012**). Syntactic Dependency-Based N-grams as Classification Features. In *Lecture Notes in Artificial Intelligence*, volume 7630. Springer Berlin Heidelberg. ISBN 978-3-642-37797-6, 1–11.
23. **Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A., & Chanona-Hernández, L.** (**2014**). Syntactic N-grams as Machine Learning Features for Natural Language Processing. *Expert Systems with Applications*, 41(3), 853–860.
24. **Socher, R., Huang, E. H., Pennington, J., Y. Ng, A., & Manning, C. D.** (**2011a**). Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems*. MIT Press.
25. **Swanson, R. & Gordon, A. S.** (**2006**). A Comparison of Alternative Parse Tree Paths for Labeling Semantic Roles. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06. ACL, Stroudsburg, PA, USA, 811–818.
26. **Ul-Qayyum, Z. & Altaf, W.** (**2012**). Paraphrase Identification using Semantic Heuristic Features.

Research Journal of Applied Sciences, Engineering and Technology, 4894–4904.

27. **Wan, S., Dras, M., Dale, R., & Paris, C.** (**2006**). Using Dependency-Based Features to Take the "Para-farce" out of Paraphrase. In *Proceedings of the Australasian Language Technology Workshop*. Sydney, Australia, 131–138.
28. **Weeds, J., Weir, D., & Keller, B.** (**2005**). The Distributional Similarity of Sub-parses. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, EMSEE '05. ACL, Stroudsburg, PA, USA, 7–12.
29. **Zhang, Y. & Patrick, J.** (**2005**). Paraphrase Identification by Text Canonicalization. In *Proceedings of the Australasian Language Technology Workshop 2005*. Sydney, Australia, 160 – 166.

Hiram Calvo obtained his PhD degree in Computer Science (with honors) in 2006 from Centro de Investigación en Computación (CIC) of the Instituto Politécnico Nacional (IPN), Mexico. His thesis was about the Spanish syntax analyzer DILUCT. He was awarded with the Lázaro Cárdenas Prize in 2006; this Prize was handed personally by the President of Mexico. He did a postdoctoral stay at the Nara Institute of Science and Technology, Japan, from 2008 to 2010. Since 2006 he is a lecturer at CIC-IPN. His research interests are lexical semantics, similarity measures, and statistical language models.

Andrea Segura-Olivares is a MSc student at CIC-IPN. Her research interests include Natural Language Processing and intelligent Human-Computer Interfaces. She obtained her BSc degree in Escuela Superior de Cómputo (ESCOM-IPN) in 2009 with honours.

Alejandro García is a MSc student at CIC-IPN. His research interests include Natural Language Processing and intelligent videogame design with natural language interaction. He obtained his BSc degree in Escuela Superior de Cómputo (ESCOM-IPN) in 2009 with honours.

Article received on 04/02/2014; accepted on 17/03/2014.