Recognizing Textual Entailment using Dependency Analysis and Machine Learning

Nidhi Sharma cs5080219@cse.iitd.ac.in
Richa Sharma anz087535@cse.iitd.ac.in
Kanad K. Biswas kkb@cse.iitd.ac.in
Indian Institute of Technology Delhi
Hauz Khas, New Delhi, India - 110016

Abstract

This paper presents a machine learning system that uses dependency-based features and lexical features for recognizing textual entailment. The proposed system evaluates the feature values automatically. The performance of the proposed system is evaluated by conducting experiments on RTE1, RTE2 and RTE3 datasets. Further, a comparative study of the current system with other ML-based systems for RTE to check the performance of the proposed system is also presented. The dependency-based heuristics and lexical features from the current system have resulted in significant improvement in accuracy over existing state-of-art ML-based solutions for RTE.

1 Introduction

Recognizing textual entailment (RTE) has aroused lot of interest in natural language research community with recent Pascal RTE challenges. RTE provides a generic evaluation framework and is useful across various applications like question-answering, information-extraction, machine translation etc.

Textual Entailment is a directional relation between text fragments (Dagan et al., 2005) which holds true when the truth of one text fragment, referred to as 'hypothesis', follows from another, referred to as 'text'. The task of recognizing textual entailment can be thought of as a classification problem to classify a given pair of sentences, text (T) and hypothesis (H), as true or false entailment as suggested by Bos and Markert (2005). Machine Learning approaches to RTE challenges have used combination of features like syntactic, semantic or lexical features. However, in most of the cases, the features used for the purpose are either large in number which makes the evaluation time consuming or are not very intuitive which makes them difficult to comprehend. In our work, we have attempted to address these two concerns.

Our approach uses a combination of dependency and lexical features to train Machine Learning (ML) classifiers. We use only 8 features that are simple and intuitive. The process of evaluating feature values is automated, thereby reducing any manual effort and intervention. The system performance has been tested over RTE1, RTE2 and RTE3 datasets. Our system shows significant improvement in accuracy over the state-of-the-art ML solutions to RTE challenges.

The paper is organized as follows. Section 2 gives a brief of the earlier work of ML based approaches for RTE. Section 3 describes our solution approach for RTE, including details on the features used and the experimental setup. We present the results and observations in Section 4, followed by conclusion in Section 5.

2 Related Work

There have been various solution approaches proposed to RTE challenges like rule-based, logical-inference based, graph-based and ML-based. Of these, applying ML algorithms to automatically learn models from training examples is an effective way to approach RTE challenges like other NLP problems.

ML-based systems often use lexical matching features (Inkpen et al. 2006, Kozareva and Motoyo 2006 and Pakray et al. 2011) such as word overlap count, word similarity, n-gram, etc, and semantic

Proceedings of NAACL-HLT 2015 Student Research Workshop (SRW), pages 147–153,
Denver, Colorado, June 1, 2015.

features such as WordNet similarity measures (Kozareva and Motoyo 2006). Inkpen et al. (2006) have achieved an accuracy of 52.85% on RTE2 dataset using lexical match and mismatch features. Bos and Markert (2005) use a combination of shallow and deep semantic features using logical inference to build a hybrid model that achieves an accuracy of 57.7 % on RTE1 dataset. They also show that using task label as feature in their model increases the overall accuracy to 61.2%. Pazienza et al. (2009) have defined a measure for textual entailment based on graph matching theory applied to syntactic dependency graphs. They perform comparison of rule-based and SVM-based approach with rule-based approach giving an accuracy of 52.45% and SVM-based approach giving an accuracy of 51.82%. Pakray et al. (2011) describe a two-way textual entailment recognition system that uses lexical features such as n-gram match, stemming etc. and syntactic features like subject comparison, subject-verb comparison, etc.

Our approach builds mainly on the works of Inkpen et al. (2006) and Pakray et al. (2011) and, improves accuracy over their work as presented in the following section.

3 Our Approach

We have developed an RTE system that takes as input sentence pairs, text (T) and hypothesis (H), and outputs an entailment decision (True/False) for each pair. The system evaluates a set of 8 different dependency and lexical features for the input sentence pairs. One of these features is a mismatch feature while the other seven are match features. For evaluating the dependency features, we have used Stanford Dependency parser (Marneffe et al., 2006) to obtain the dependency relations present in the sentences. We generate a structured representation for both text and hypothesis using their respective dependency relations. This structured representation is used to evaluate six of the eight lexical and the syntactic features. Structured representation proves to be an effective representation of the sentence for calculating feature values.

We first present a brief overview of the structured representation of the sentences before discussing the features used in the feature vector to develop classifiers using ML algorithms.

3.1 Structured Representation

The Stanford dependencies describe the grammatical relationships in a sentence. Each dependency is a tuple consisting of a pair of words in a sentence and the relation that links them. A dependency is represented as follows:

reln(govVal,depVal)

where,

reln is the dependency relation
depVal is the dependent value
govVal is the governing value

The structured representation is generated by using the dependency tags and converting them to a slot-filler frame-based structure. The entities extracted from the dependency relations are:

a) Subject: The dependencies tagged as *nsubj* (Nominal Subject), *nsubjpass* (passive nominal subject), *csubj* (clausal subject), *csubjpass* (passive clausal subject) and *xsubj* (controlling subject) are used to extract the words acting as subject in the sentence.

b) Subject Modifier: The dependency tags *advmod* (adverbial modifier), *amod* (adjectival modifier), *appos* (appositional modifier), *nn* (noun compound modifier) and *npadvmod* (noun phrase as adverbial modifier) are used to identify modifiers. Each dependency relation is returned as a pair of (governing value, dependent value). If for a given modifier relation, the governing value is a subject in the sentence, then the dependent value acts as the subject modifier.

c) Object: The Stanford parser returns *dobj* (direct object), *iobj* (indirect object) and *pobj* (prepositional object) as tags for different objects. We include all these in the frame entity 'object'.

d) Object Modifier: The process to extract object modifier is similar to the one used for Subject Modifier except that if the governing value in the modifier relation is an object in the sentence, the dependent value acts as the object modifier.

e) Verb: The dependency tagged as root is generally the main verb of the sentence. The tags *cop* (copula), *ccomp* (clausal complement) and *xcomp* (open clausal complement) also list the verbs in the sentence.

f) Verb Complements: In some cases, the dependencies tagged as *root*, *xcomp* or *ccomp* (clausal complement) contain noun instead of verb. The dependent value is then listed as a verb complement. The tags *acomp* (adjectival complement),

pcomp (prepositional complement), *advcl* (adverberial clause modifier) and *vmod* (verbal modifier) also contains dependency values that complement the verb.

g) Negation: The parser uses the tag *neg* (negation modifier) to identify negation words (such as, not, don't, etc.) in the sentence. The governing value of this dependency contains the word (usually verb) it negates. We store this value with the negation word for negation frame entity.

h) Number: The dependency tagged as *num* (numeric modifier) contains a numeric value and the noun-phrase that it modifies. We store both the number and the entity it modifies under this label.

The generation of the frame-based structured representation is illustrated using statement S1 and this structured representation is shown in Table 1.

S1: *A two-day auction of property belonging to actress Katharine Hepburn brought in 3.2 million pounds.*

Label	Value
Subject	Auction
Subject Modifier	two-day
Object	property, Hepburn, pounds
Object Modifier	Actress
Verb	Brought
Verb Complements	Belonging
Negation	-
Number	3.2 million (pounds)

Table 1: Structured Representation for S1

3.2 Features

After obtaining a structured representation, we evaluate the following features (i) to (viii). While features (i) and (ii) have been borrowed from previous work (Inkpen et al. 2006, Kozareva and Motoyo 2006, Pakray et al. 2011), the features (iii), (iv) and (iv), present significant modifications to features used by researchers (Inkpen et al. 2006, Molla 2003 and Pakray et al. 2011) in the past. The features (vi), (vii) and (viii) are new features contributing to our feature set. The *dependency overlap* and *word overlap* features do not require structured representation for evaluation.

(i) Word Overlap

This feature is a ratio of the count of directly overlapping words between text and hypothesis to the count of words in hypothesis, after removal of stop words. A direct word count also takes care of the overlapping named entities. This feature is a significant contributor to entailment. The overlap is evaluated as follows:

$$wordOverlap = \frac{countTnH}{countH}$$

where,
countTnH = number of common words in text and hypothesis after stop word removal
countH = total number of words in hypothesis after stop word removal

(ii) Negation Check

This feature checks if a verb in hypothesis has been negated in text or vice-versa. Negation can be explicit in the form of keywords, such as 'not', 'can't', 'don't', etc. or it can be implicit in the form of antonym or negative sense of the verb. We capture explicit as well as implicit negation check through the structured representation of the sentence. In order to identify if the antonym of a verb (non-negated) in hypothesis is present in text or vice-versa, we first identify the root form of the verbs present in text as well as hypothesis using Wordnet[1]. The root form of the verbs is then checked for antonym (or negative sense) relationship by using VerbOcean[2].

This is a binary feature assuming a value of 1 for the presence of negation, either explicit or implicit and, it remains 0 otherwise. For example, consider the following text-hypothesis pair:

T: *The Philippine Stock Exchange Composite Index rose 0.1 percent to 1573.65*

H: *The Philippine Stock Exchange Composite Index dropped.*

In this example, the verbs 'rose' and 'dropped' are converted to their root forms 'rise' and 'drop' respectively and found to have an antonym relation (rise [opposite-of] drop) in VerbOcean.

(iii) Number Agreement

This is a binary feature to check if the numeric modifiers of the same governing entities are in agreement in text-hypothesis pair. We use structured representation to evaluate this feature. The feature takes a value of 1 for number agreement and 0 otherwise. We illustrate number agreement

[1]http://projects.csail.mit.edu/jwi/
[2] http://demo.patrickpantel.com/demos/verbocean/

using the pair T1-H1 and number disagreement with the help of pair T2-H2 as follows:

T1: *The twin buildings are 88 stories each, compared with the Sears Tower's 110 stories.*

H1: *The Sears Tower has 110 stories.*

T2: *A small bronze bust of Spencer Tracy sold for £174,000.*

H2: *A small bronze bust of Spencer Tracy made £180,447.*

(iv) Dependency Overlap

Dependency overlap has been considered as a good approximation to sentence meaning in context of question-answering problem by Molla (2003). We have borrowed the same idea to approximate the entailment relationship between text and hypothesis. The dependency relations returned by the Stanford parser consist of a pair of words from the sentence that are related. We count such similar pairs irrespective of the relation binding them. The value of the feature is computed as:

$$depOverlap = \frac{countTnH}{countH}$$

where,

countTnH = number of overlapping dependency pairs in text and hypothesis and,

countH = total number of dependencies in hypothesis

Considering an example:

T: *His family has steadfastly denied the charges*

H: *The charges were denied by his family*

Dependency list for T is:

[*poss(family-2, His-1), nsubj(denied-5, family-2), aux(denied-5, has-3), advmod(denied-5, steadfastly-4), root(ROOT-0, denied-5), det(charges-7, the-6), dobj(denied-5, charges-7)*]

Dependency list for H is:

[*det(charges-2, The-1), nsubjpass(denied-4, charges-2), auxpass(denied-4, were-3), root(ROOT-0, denied-4), poss(family-7, his-6), agent(denied-4, family-7)*]

This example has five overlapping dependency pairs, namely: *the-charges, denied-charges, ROOT-denied, his-family* and *denied-family*. We evaluate dependency overlap for this example as follows:

$$depOverlap = \frac{countTnH}{countH} = \frac{5}{6} = 0.833$$

(v) Syntactic Role Match

This feature is set to 1 if the (subject, object, verb) tuple in text matches the (subject, object, verb) tuple in hypothesis. The subject and object are matched directly whereas the verbs are matched after extracting their root forms from Wordnet and using the 'similar' relation from VerbOcean.

Similar feature has been used in Pakray et al.'s (2011) approach, wherein they have considered matching pairs of subject-verb, verb-object, subject-subject and object-object. However, the semantics of any sentence are governed by subject, verb and the object, if present. Our feature differs in the sense that a value of 1 is assigned for matching of the subject, object and the verb altogether; else its value remains 0. For example:

T: *Israeli Prime Minister Ariel Sharon threatened to dismiss Cabinet ministers who don't support his plan to withdraw from the Gaza Strip.*

H: *Israeli Prime Minister Ariel Sharon warned to fire cabinet opponents of his Gaza withdrawal plan.*

In this example, the subject in both T and H is *Ariel Sharon*, the direct object in T is *plan* whereas the direct object in H, is *opponents* but H has *plan* as the prepositional object and so we consider it as an object agreement. The verbs 'threaten' in T and 'warn' in H are similar as inferred from VerbOcean. Therefore, the value of syntactic-role match feature for the above-mentioned text-hypothesis pair is 1. In contrast, following Pakray et al.'s (2011) approach, the value of Wordnet-based subject-verb feature is 0.5 instead of 1 and the value of Wordnet-based verb-object feature is 0 due to mismatch in direct object.

(vi) Complement Verb Match

The sentences are not always simple and apart from main action-verbs, there can be entailment relationship due to complementing verb or clausal components. This feature performs a semantic match of root form (derived from Wordnet) of such verbs of text and hypothesis using VerbOcean. In addition, it also checks if the acting verb of hypothesis matches the acting verb or verb complement of the text and vice-versa. Let us consider an example to understand such pairs:

T: *Officials said Michael Hamilton was killed when gunmen opened fire and exchanged shots with Saudi security forces yesterday.*

H: *Michael Hamilton died yesterday.*

The main verb in T is 'said' while the main verb in H is 'died' and these verbs do not match. However, 'killed' is a clausal complement in T which is similar to the verb 'died' in H. Thus, a match results in this case assigning a value of 1 to the feature else the value of the feature would be 0.

(vii) Modifier Relation

In this feature, we check if the subject-object pair of hypothesis appears as subject-subject modifier or object-object modifier pair in the text. It is also a binary feature assuming a value of 1 for match and 0 for mismatch. For example:

T: *Antonio Fazio, the Bank of Italy governor, engulfed in controversy.*

H: *Antonio Fazio works for the Bank of Italy.*

In T, '*Antonio Fazio'* is the subject and '*Bank of Italy governor'* is the appositional modifier of the subject. In H, '*Antonio Fazio*' is the subject and '*Bank of Italy*' is the object. Therefore, a match occurs and the value of feature assigned is 1.

(viii) Nominalization

This features checks for nominal forms of the verbs as there can be correspondence between text and hypothesis owing to nominal verbs. We check if the nominal form of a verb in hypothesis acts as object in the text or the nominal form of verb in text acts as object in hypothesis. If a match is found, then we assign 1 to this feature else we assign 0. Following pair presents one such example:

T: *Satomi Mitarai died of blood loss.*

H: *Satomi Mitarai bled to death.*

In this example, the verb 'bled' in H has its noun-form 'blood' in T and the verb 'died' in T has its noun-form 'death' in H.

3.3 Experimental Setup

The system performance is evaluated by conducting experiments on RTE1, RTE2 and RTE3 datasets. The RTE1 dataset consists of 567 sentences pairs (T and H) in the development set and 800 sentence pairs in the test set. These sets are further divided into seven subsets, namely: Information Retrieval (IR), Comparable Documents (CD), Question Answering (QA), Information Extraction (IE), Machine Translation (MT) and Paraphrase Acquisition (PP). The RTE2 and RTE3 datasets contain 800 sentence pairs each in their development as well as test sets. Both the development and test sets of RTE2 and RTE3 are subdivided into four tasks, namely: IE, IR, QA and SUM (summarization).

We have conducted experiments with different ML algorithms including Support Vector Machines (SVM), Naïve Bayes and Decision Trees (DT) using Weka[3] tool. For each of the RTE datasets, respective training set has been used while experimenting with corresponding test-set. We have also performed task based analysis for RTE1 dataset. Following section summarizes the observations of our experiments.

4 Results

Table 2 presents the results achieved with 67% split evaluation of the classifiers on each of the development (training) datasets:

Classifier	Accuracy	Precision	Recall
RTE - 1			
NB	59.28	57.8	68.6
SVM	**67.02**	63.3	80.9
DT	66.07	64	73.6
RTE - 2			
NB	60.62	62.2	54.3
SVM	**65.75**	67	62
DT	63.0	60.7	73.8
RTE - 3			
NB	64.75	68.2	59.2
SVM	66.62	66.4	71.4
DT	**67.87**	67	74

Table 2: Validation of system on development sets

As evident from table 2, highest accuracy is achieved with DT algorithm and SVM with RBF kernel. DT learns very fast and identifies strong relationship between input and target values (Ville, 2006). In our case, DT turned out to be efficient and fast learners to identify relationship between the feature vectors and the expected entailment results. For SVM, though it is not guaranteed which kernel performs better in a situation, RBF kernel is generally more flexible than the linear or polynomial kernels as it can model a high dimensional feature space with minimum error. The observations with these algorithms are strengthened by the test-set results as presented in table 3.

We have also experimented by using task label as a feature in our system as Bos and Markert

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

(2005) experimented with their system. Like Bos and Markert's (2005) observation, we also found that the system performance increases with DT algorithms in contrast to other ML classifiers. Table 4 shows our system's performance on RTE1, RTE2 and RTE3 datasets using DT algorithm.

Classifier	Accuracy	Precision	Recall
RTE – 1			
NB	57.62	56.4	67.5
SVM	57.25	57.6	55.3
DT	**60.12**	60.3	68.7
RTE – 2			
NB	59.12	60.9	60
SVM	59.62	60	61
DT	**59.87**	57.5	73.2
RTE – 3			
NB	60.62	62.1	63.9
SVM	62.12	61.5	69.75
DT	**62.75**	62	71

Table3: Performance of system on test sets

DataSet	Accuracy	Precision	Recall
RTE1	**61.25**	61.7	57.7
RTE2	**60.41**	62.8	60.3
RTE3	**64.38**	62	78

Table 4: System Performance - Task label as Feature

For task-based analysis, we experimented with the tasks of RTE1 dataset separately. We present the comparative study of the accuracy achieved by our system with the SVM-based solution of Pazienza et al. (2005) and DT-based solution of Bos and Markert (2005) in table 5. The improvement in accuracy by our system is reflected in table 5.

Task	Pazienza et al. (2005)	**Our System (SVM)**	Bos & Markert (2005)	**Our System (DT)**
IE	49.17	**59.16**	54.2	**55.83**
IR	48.89	**71.33**	62.2	**67.51**
QA	45.74	**63.84**	56.9	**60.76**
MT	47.9	**62.5**	52.5	**58.33**
RC	52.14	**62.0**	50.7	**61.3**
CD	64.43	**83.46**	70.0	**81.04**
PP	50.0	**78.03**	56	**75.75**

Table 5: Task-based performance comparison for RTE1 test set

We carried out a comparative study of our system with other ML-based systems for RTE to check the performance of our system. The observations from this comparative analysis of our system with relevant related systems for RTE along with the feature counts (FC) used by the respective systems in presented in table 6. The comparative study indicates significant improvement in accuracy of our system over most of the existing state-of-art ML-based solutions for RTE except for few solutions only.

Accuracy	FC	RTE 1	RTE 2	RTE 3
(Bos&Markert, 2005)[4]	> 8	57.7	-	-
(Inkpen et al., 2006)	26	-	58.25	-
(Kozareva & Montoyo, 2006)	17	-	55.8	-
(Pakray et al., 2011)	16	53.7	59.2	61
(MacCartney et al., 2006)	28	59.1	-	-
(Hickl et al., 2006)	12	-	**65.25**	-
(Adams et al., 2006)	13	-	-	**67**
Ours	**8**	**60.12**	**59.87**	**62.75**

Table 6: Comparison of accuracy of our system with other systems

5 Conclusion

As the results indicate, our dependency-based heuristics and lexical features have resulted in significant improvement in accuracy of RTE1, RTE2 and RTE3 datasets. DT outperforms other classifiers with only 8 features that are syntactic and lexical in nature. SVM classifier shows comparable performance with the RBF kernel. The features are simple and intuitive; easy to comprehend and evaluate. The task-based performance for RTE1 dataset shows improved performance as compared to the similar study by Pazienza et al. (2005) and by Bos and Markert (2005). We intend to identify more syntactic and semantic features in future and improve upon and, experiment with them to refine the results further.

[4]Authors have used 8 deep semantic feature and some shallow lexical features, count of which is not clear from the paper. Therefore, we are considering their feature-count to be more than 8

References

Andrew Hickl, Jeremy Bensley, John Williams, Kirk Roberts, Bryan Rink and Ying Shi. 2006. Recognizing Textual Entailment with LCC's Groundhog System. In *Proceedings of the Second PASCAL Challenges Workshop on Recognizing Textual Entailment*.

Barry de Ville. 2006. *Decision Trees for Business Intelligence and Data Mining*. SAS Enterprise Miner.

Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer and Christopher D. Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of North American Chapter of ACL (NAACL-2006*.

Diana Inkpen, Darren Kipp, and Vivi Nastase. 2006. Machine Learning Experiments for Textual Entailment. In *Proceedings of the Second Challenge Workshop Recognizing Textual Entailment*: 10-15, Italy.

Diego Molla. 2003. Towards semantic-based overlap measures for question answering. In *Proceedings of the Australasian Language Technology Workshop 2003,* Australia.

Fabio M. Zanzotto, Marco Pennacchiotti and Alessandro Moschitti. 2009. A machine learning approach to textual entailment recognition. *Natural Language Engineering*, 15(4): 551-582.

Ido Dagan, Oren Glickman and Bernardo Magnini. 2005. The PASCAL Recognizing Textual Entailment Challenge, In *Proceedings of the First PASCAL Recognizing Textual Entailment Workshop*.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference, In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*: 628-635.

Maria T. Pazienza, Marco Pennacchiotti and Fabio M. Zanzotto. 2005. Textual Entailment as Syntactic Graph Distance: a Rule Based and a SVM Based Approach. In *Proceedings of first PASCAL RTE challenge*:528—535.

Marie-Catherine de Marneffe, Bill MacCartney and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *LREC 2006*.

Partha Pakray, Alexander Gelbukh and Sivaji Bandyopadhyay. 2011. Textual Entailment using Lexical and Syntactic Similarity. *International Journal of Artificial Intelligence & Applications (IJAIA)*, 2(1): 43-58.

Rod Adams, Gabriel Nicolae, Cristina Nicolae and Sanda Harabagiu. 2007. Textual Entailment through Extended Lexical Overlap and Lexico-Semantic Matching. In *Proceedings of the Third PASCAL Challenges Workshop on Recognizing Textual Entailment*.

Zornitsa Kozareva and Andrés Montoyo. 2006. MLEnt: The Machine Learning Entailment System of the University of Alicante. In *Proceedings of the Second Challenge Workshop Recognizing Textual Entailment*: 17-20.