Natural Language Engineering **23** (4): 507–534. © Cambridge University Press 2016
doi:10.1017/S1351324916000176

A transformation-driven approach for recognizing textual entailment†

ROBERTO ZANOLI[1] and SILVIA COLOMBO[2]
[1]*Human Language Technology, Fondazione Bruno Kessler, 38123 Trento, Italy*
e-mail: zanoli@fbk.eu
[2]*Edinburgh University School of Informatics, 11 Crichton St, Edinburgh EH8 9LE, UK*
e-mail: s1132418@sms.ed.ac.uk

(*Received 11 May 2015; revised 1 May 2016; accepted 3 May 2016; first published online 16 June 2016*)

Abstract

Textual Entailment is a directional relation between two text fragments. The relation holds whenever the truth of one text fragment, called Hypothesis (H), follows from another text fragment, called Text (T). Up until now, using machine learning approaches for recognizing textual entailment has been hampered by the limited availability of data. We present an approach based on syntactic transformations and machine learning techniques which is designed to fit well with a new type of available data sets that are larger but less complex than data sets used in the past. The transformations are not predefined, but calculated from the data sets, and then used as features in a supervised learning classifier. The method has been evaluated using two data sets: the SICK data set and the EXCITEMENT English data set. While both data sets are of a larger order of magnitude than data sets such as RTE-3, they are also of lower levels of complexity, each in its own way. SICK consists of pairs created by applying a predefined set of syntactic and lexical rules to its T and H pairs, which can be accurately captured by our transformations. The EXCITEMENT English data contains short pieces of text that do not require a high degree of text understanding to be annotated. The resulting AdArte system is simple to understand and implement, but also effective when compared with other existing systems. AdArte has been made freely available with the EXCITEMENT Open Platform, an open source platform for textual inference.

1 Introduction

Textual Entailment (*TE*) is a directional relation between two text fragments, one called Text (T) and the other Hypothesis (H). The relation holds if, typically, a

†This work has been supported by the EC-funded project EXCITEMENT (FP7-ICT-287923) and in part done during the internship of the second author at Fondazione Bruno Kessler (FBK) in Italy. We would like to thank the EOP developers for making the EOP publicly available and supporting us; in this regard, many thanks to Vered Shwartz and Tae-Gil Noh; thanks to Lili Kotlerman and Vivi Nastase for creating and distributing the EXCITEMENT data set. Thanks to Thomas Proisl and his team for testing their system (SemantiKLUE) on EXCITEMENT. Finally, a special thanks goes to Luisa Bentivogli for providing us with the new release of the SICK data set, and Alberto Lavelli for his comments and valuable suggestions.

human reading T would infer that H is most likely true. As an example, consider the following text fragments where, according to the definition of entailment, only H1 can be inferred from T:

T: *Hubble is a telescope that revolves around Earth*
H1: *Hubble is a scientific instrument that orbits Earth*
H2: *Hubble was launched into orbit in 1990*

In 2004, Recognizing Textual Entailment (RTE) (Dagan, Glickman and Magnini 2005) was proposed as a generic task to capture such inferences. In its original formulation, given two text fragments, the task consists in establishing whether the meaning of the Hypothesis can be inferred from the Text. Many Natural Language Processing applications can benefit from this taskshort (Dagan *et al.* 2009). In summarization, for example, a summary should be entailed by the text; in question answering and information retrieval, the answer for a query must be entailed by the supporting snippet of text.

Using machine learning techniques for RTE is limited by the amount of available labeled data and researchers have had to consider the issue of how the dimensionality of the feature space should vary with respect to the sample size. Intuitively, an imbalance between the number of features used and the number of labeled samples can make classification more difficult. As a consequence, when working with data sets as small as RTE-3 (Giampiccolo *et al.* 2007), many researchers use heuristic feature selection methods in order to reduce the feature space and obtain good feature combinations. The solutions that have been proposed, however, while facilitating support with such data sets, might not work well with new data sets that consist of many samples, for which it may be convenient to use a greater number of more complex features.

In this paper, we present AdArte (A transformation-driven approach for RTE), a system that combines syntactic transformations and machine learning techniques to establish the entailment relations in T–H pairs. Unlike other existing methods, AdArte is designed to fit well with a new kind of data sets that are larger but less complex than data sets used in the past.

The algorithm has been tested on two data sets: the SICK data set, which was used at SemEval-2014 Task#1, and the EXCITEMENT English data set, which is a new data set composed of email feedback sent in by the customers of a railway company. While both data sets are of a larger order of magnitude (higher number of annotated samples) than RTE-3, they also present lower levels of complexity, in different ways.

SICK consists of pairs created by applying a predefined set of syntactic and lexical rules to its T and H pairs. The rules can preserve the meaning (e.g., turning active sentences into passive), create contradiction (e.g., replacing words with semantic opposites) or create unrelated pairs (e.g., switching and mixing modifiers). To clarify these differences between SICK and RTE, consider the pair T:*A man is driving a car*, H:*The car is being driven by a man* extracted from SICK, and the one extracted from RTE: T:*US Steel could even have a technical advantage over Nucor since one new method of steel making it is considering, thin strip casting, may produce* *higher quality steel than the Nucor thin slab technique*, H:*US Steel may invest in strip casting*. It is clear that in the case of the RTE pair syntax alone is not sufficient to determine entailment as in the case of the SICK pair, and that some degree of text understanding is required.

The use of such a procedure for the creation of SICK means that there is a correspondence between the rules used to produce a pair (e.g., creating contradictions) and the entailment label assigned to the pair (e.g., contradiction). AdArte learns a set of rules equivalent to those above in order to make an inference judgment. In short, it models the entailment relation as a classification problem where the individual T–H pairs are first represented as a sequence of syntactic operations, called transformations, needed to transform the Text (T) into the Hypothesis (H), and then used as features in a supervised learning classifier that labels the pairs as positive or negative examples. The transformations are calculated by applying tree edit distance on the dependency trees of the T–H pairs, while some knowledge resources, such as WordNet, are used for recognizing cases where T and H use different textual expressions (e.g., *home* versus *house*) that still preserve entailment. For classification, we have used Weka (Hall *et al.* 2009), a collection of machine learning algorithms that enabled us to try different classifiers, like Random Forests and Support Vector Machines (SVM).

Unlike SICK, the EXCITEMENT English data set has not been created by applying any specific kind of rules to its T–H pairs. However, the data set still maintains a low level of complexity due to its short pieces of text, which do not require a high degree of text understanding in order to be annotated.

AdArte is simple to understand and implement but also effective when compared with the systems that took part in SemEval-2014 Task#1 (Marelli *et al.* 2014a), and the ones distributed as part of the EXCITEMENT Open Platform (EOP) (Padó *et al.* 2013; Magnini *et al.* 2014), which is an open source software platform for RTE relations. AdArte has been made available with the EOP for research use and evaluation.

The paper is organized as follows: Section 2 discusses the related work, while Section 3 describes the data sets used for the system evaluation. Section 4 introduces the transformations and Section 5 goes on to describe the system architecture. Section 6 moves to the experimental part and finally Section 7 presents the conclusions.

2 Related work

TE systems tend to follow the same fundamental approach, which consists in representing the meaning of the Text (T) and the Hypothesis (H) at a certain level and checking if the representation of the meaning of H is contained in that of T. Generally, this is done in one of two ways: (i) purely lexical approaches for which, in the simplest case, the likelihood of entailment is based on lexical overlap between the words of the Text and those of the Hypothesis, (ii) deeper proof-theoretic approaches which use deductive proof mechanisms to perform the entailment judgment. The more successful systems for RTE can be classified into several different categories based on the model they use. In this section, we divide the models into the following categories: alignment-based models, transformation-based models and proof-theoretic models. For a more detailed analysis of the principal techniques that have been developed so far, see Dagan *et al.* (2013).

The best performing system at the SemEval-2014 Task#1 was the system proposed by the University of Illinois (Lai and Hockenmaier 2014). It combines non-compositional features (e.g., word overlap and alignment between the fragments), compositional features and denotational similarities.

Edit distance models can be viewed as transformation-based models with very simple rules. They were first adopted for RTE in EDITS (Kouylekov and Magnini 2005). In this approach, the distance values calculated on the trees of the T–H pairs in the development set are used to compute a threshold value for separating the entailment pairs from the non-entailment ones. If the distance value between T and H in an unannotated pair is below or equal to the calculated threshold, then there exists entailment between T and H; the relation is non-entailment otherwise. As opposed to this method, we use the transformations produced by tree edit distance as features to create a feature vector for every T–H pair.

Heilman and Smith (2010) suggested an approach using transformations which has some superficial similarities to ours; they used tree edit models to represent sequences of tree transformations and then applied the logistic regression classification model by Hastie, Tibshirani and Friedman (2001) on the labels and features of the extracted sequences. Our method differs from their approach in three important aspects: (i) They extended the three basic edit operations with additional and more complex ones, like the operation that involves moving whole subtrees. Next, to efficiently search for a reasonable sequence of transformations, they used *greedy best-first search* and the computationally expensive *Tree Kernel-based heuristic function*. In contrast, we use tree edit distance to extract three simple types of transformations (i.e., insertion, deletion, substitution), where the number of produced transformations is the minimum number of transformations required to transform one tree into another. (ii) They selected a well-defined set of 33 features that they considered to be the most promising with the specific data set. Instead, we use all the transformations produced, without any specific manual or semi-automatic feature selection dependent on the data set. In addition, the feature set used is homogeneous: all the features represent a type of transformation. (iii) While they used logistic regression for classification, we used other classifiers that performed better on our data sets and with our feature sets. Our method has the following advantages with respect to theirs: it appears simpler to implement and easier to adapt to new data sets, and is more efficient in terms of computational time. The features can be extracted by using one of the many available implementations of tree edit distance and no feature selection is required; there are no constraints to be set or integrated. In contrast, we think that their method would outperform our method when tested on data sets like RTE-3. In fact, the high number of features produced by our method, combined with the low number of available labeled samples, would make classification more difficult in this case.

Transformation-based models, which are a step forward toward proof-theoretic models, were suggested by Harmeling (2009) and Stern and Dagan (2012). Harmeling's proposal was a system for TE based on probabilistic transformation. Supposing that there may be multiple transformation sequences (i.e., derivations) that derive a given Hypothesis from a given Text, the derivation with the highest probability should be selected. An entailment label for such a T–H pair can be determined by selecting a threshold (such as $p \geq 0.5$) above which entailment is considered to hold. Our method differs in three ways from their work: (i) Harmeling defined a fixed set of operations, including syntactic transformations, WordNet-based substitutions, and more heuristic transformations such as adding or removing a verb or even substituting a named entity. In contrast, in our approach, the transformations are extracted automatically and their number and type depends on the syntactic structure of the given pairs. (ii) He applies the transformations in a predefined order, while we execute them in an order so as to produce a minimum number of transformations. (iii) In Harmeling's approach, the probability of each predefined transformation is estimated from the development set, and the probability that a transformation sequence is consistent with T (i.e., the entailment holds) is defined as the product of the probability of each transformation in the sequence. Instead, our approach uses a vector space model for representing the T–H pairs as vectors of transformations, and machine learning-based classifiers to assign an entailment label to such vectors.

In Stern and Dagan's proposal, they apply the transformations derived from knowledge and linguistic resources, such as WordNet and Wikipedia, to preserve the meaning of the text, as follows: when a transformation is applied on a text T, it transforms it into a new text, T′. The goal is that the meaning of T′ will be entailed from T. Then, they estimate the validity of every transformation and, consequently, of every sequence of transformations, and decide whether T entails H based on this estimation. There are two main differences compared to our method: (i) in their method, the transformations are extracted from external resources whereas we extract them from the data set. (ii) They apply the calculated transformations as a sequence of operations in order to determine whether H can be obtained from T while we use the transformations as features for the classifier. We believe our method to be more reliant on the data set for capturing the relevant syntactic phenomena that determine whether an entailment relation holds. Learning from these transformations requires a number of annotated examples which might not be available in some application domains. Their method appears to be able to learn from a smaller number of examples, and might be more suitable for domain adaptation, when the data distribution in the test domain is different from that in the training domain. However, unlike our approach, which allows a relatively easy adaptation to new languages (provided that annotated data is available), the portability of their approach to languages other than English is uncertain, as the same linguistic resources might not be available for many other languages.

More formal proof-theoretic models encode facts in a knowledge base using a formal representation, such as propositional or first-order logic, and apply rules of inference to determine the set of facts that can be derived from the knowledge base.

Systems that use theorem-proving as their conceptual basis were suggested by Bos and Markert (2005) and MacCartney (2007).

Finally, it is worth mentioning a recent study by Bowman *et al.* (2015) using neural networks. Training a neural network on benchmarks such as SICK results in poor performance due to the even larger amount of data required. To face this issue, they trained the neural networks in a transferable way: they used the tuned parameters calculated on a new and large data set (i.e., the Stanford Natural Language Inference data set) to initialize the parameters of a model on the target SICK data set. Even though the measured values are slightly divergent from the state-of-art, they obtained the best performance yet reported for an unaugmented neural network model.

3 Data sets

We used two data sets for the system evaluation: SICK and EXCITEMENT. SICK is the data set used at SemEval-2014 Task#1. Its new release, of which we had a first preview, includes additional information about the syntactic and lexical rules used for creating its T–H pairs. This allowed us to evaluate our method while also considering the different phenomena taking place within the data set (e.g., evaluation on the T–H pairs where H has been created by the annotators transforming T into passive rather than replacing its words with semantic opposites). On the other hand, the EXCITEMENT English data set is a new data set containing email feedback from the customers of a railway company. Details of these data sets are given in Sections 3.1 and 3.2.

3.1 SICK data set

The SICK data set (Marelli *et al.* 2014b) was used at SemEval-2014 Task#1 for the subtasks of: (i) relatedness (i.e., predicting the degree of semantic similarity between two sentences), and (ii) entailment (i.e., detecting the entailment relation between two sentences).

SICK consists of 9,840 English annotated T–H pairs: 4,934 as part of the development set and 4,906 in the test set. Examples of such T–H pairs can be seen in Table 1.

To produce the data set, the annotators randomly chose 750 images from the 8K ImageFlickr data set[1] and then sampled two descriptions from each of them, while another 750 sentence pairs were extracted from the SemEval-2012 STS MSR-Video Descriptions data set.[2] Next, they applied a fours-step process consisting of:

- Normalization: from the original sentences (S0), new sentences (S1) were produced by excluding or simplifying instances that contained lexical, syntactic or semantic phenomena such as named entities, dates, numbers, multi-word expressions or normalizing the tense.

[1] http://nlp.cs.illinois.edu/HockenmaierGroup/data.html
[2] http://www.cs.york.ac.uk/semeval-2012/ task6/index.php?id=data

Table 1. *SICK: examples of annotated T–H pairs*

Pair	Entailment relations
T: Two dogs are fighting H: Two dogs are wrestling and hugging	Neutral
T: Three boys are jumping in the leaves H: Three kids are jumping in the leaves	Entailment
T: There is no biker jumping in the air H: A lone biker is jumping in the air	Contradiction

- Expansion: each of the normalized sentences (S1) was expanded in order to obtain (i) a sentence with a similar meaning (S2) which means the overall meaning of the sentence does not change, (ii) a sentence with contrasting meaning (S3) and (iii) a sentence that contains most of the same lexical items, but has a different meaning (S4).
- Pairing: S2, S3 and S4 were paired with the S1 sentence from which they were created. Then, S2, S3 and S4 were paired with the other S1 sentences in the original pair. Finally, sentences from different original pairs S0 were randomly paired.
- Annotating: the pairs produced in the previous step were finally annotated with one of three possible gold labels: entailment, contradiction (i.e., the negation of the conclusion is entailed from the premise) and neutral (i.e., the truth of the conclusion cannot be determined on the basis of the premise).

The fact that this procedure has been applied for the creation of the data set means that there is a correspondence between the syntactic and lexical rules applied to produce a pair and the entailment label assigned to the pair. The entailment label is mostly assigned in the case of S1–S2 pairs (similar meaning), the contradiction label in the case of S1–S3 pairs (contrast/contradiction), and the neutral label in the case of S1–S4 pairs (lexical overlap only). However, the data set also contains a relatively high proportion of S1–S3 pairs (and, even if to a lesser extent, of S1–S2 pairs) labeled neutral. Inspection of the neutral pairs reveals a significantly higher incidence of pairs of sentences with subjects with indefinite papers. For example, T:*A couple is not looking at a map* does not contradict H:*A couple is looking at a map*, because one could imagine two different couples, one who looks at a map, while the other one does not. In contrast, T:*A person on a bike is not in the air near a body of water* contradicts H:*A person on a bike is in the air near a body of water*, because in this case, the people mentioned in the two sentences were thought to be the same person. As it is discussed in Section 6.5, this fact impacts the quality of the system's output. Another aspect that is worth highlighting is that the data set presents some 'flaws' that both non-compositional and compositional systems at SemEval-2014 Task#1, took advantage of. In particular, one of them reports that, just by checking the presence of negative words, one can detect 86.4 per cent of the contradiction pairs, and by combining Word Overlap and antonyms one can detect 83.6 per cent of neutral pairs and 82.6 per cent of entailment pairs. Another participant reports

Table 2. *EXCITEMENT: examples of annotated T–H pairs*

Pairs	Entailment relations
T: Food could be better H: Better food would be good	Entailment
T: I always get the same tasteless food H: Food is unhealthy	Non-entailment

that a surprisingly helpful feature was the sentence pair's ID. In this respect, our approach does not make use of any rule or feature created ad hoc for that purpose. The features used and their number is determined automatically by our method according to the characteristics of the data set. For example, as mentioned before, there is no doubt that the presence of a relation between a negation word and the word it modifies is important in detecting the contradiction pairs; and it is equally true that our method can recognize and exploit such a phenomenon for the pairs classification. However, this must be regarded as a characteristic of our approach that lies in the ability to learn from the various lexical and syntactic phenomena which arise within the data set. This happens without the need for hand-written rules, which might be designed to work on a specific data set, but might fail when attempting to annotate new and different data sets.

3.2 EXCITEMENT english data set

In order to verify the applicability of our method to a data set other than SICK, and to compare our results with several other existing systems, we used the EXCITEMENT english development data set (Kotlerman *et al.* 2015), which is freely available from the EOP web site.[3] EXCITEMENT contains pieces of email feedback sent by the customers of a railway company where they state reasons for satisfaction or dissatisfaction with the company. Table 2 shows an example of T–H pairs annotated with one of the two possible entailment relations in the data set, i.e., entailment and non-entailment. Working with this data set is different from annotating the RTE data sets in many aspects. As regards, the preprocessing, capitalization and punctuation marks in the data set are often missing. Moreover, it contains a lot of imperatives (e.g., improve the food), and noun phrases (e.g., Passageway too narrow), which may deteriorate the performance of methods that use deep linguistic analysis annotations, as our method does.

The data distribution was generated manually by its annotators by organizing the emails into groups based on the nineteen different topics they relate to, like *food choice* and *Internet*, and then splitting them to generate four different data sets, namely mix-balanced, mix-unbalanced, pure-balanced and pure-unbalanced.[4] For these sets, the distinction between balanced and unbalanced data sets lies in

[3] `https://github.com/hltfbk/EOP-1.2.1/wiki/Data-Sets`
[4] Only the development set is available at the time we are writing.

whether the data set contains a comparable number of examples per category or a large amount of examples from only one category. The terms mix and pure refer to how the T–H pairs are distributed among the data sets. In the case of the Mix data set, the T–H pairs referring to a specific topic (e.g., food choice) are equally distributed between the training and test data set (i.e., both the training and test contain examples of food choice). In contrast, in the pure data set, the training and the test set contain pairs referring to different topics (e.g., the training could contain the examples for food choice, whereas the test, the ones for Internet).

The reason for creating this split is that it sheds light on how different RTE systems perform under different conditions (e.g., a system might perform better on balanced data sets, while another system could be insensitive to that).

4 Transformations

Transformations are elementary edit operations needed to transform one text into another. The most widely used edit operations are deleting, replacing and inserting pieces of text. In the context of TE, the transformations of a T–H pair are the edit operations needed to transform T into H. For example, given the following T–H pair extracted from SICK:

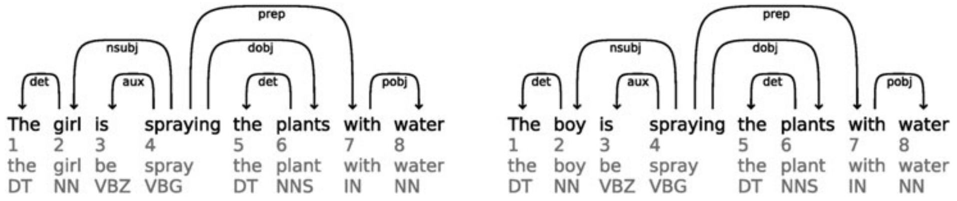
label: NEUTRAL
T:*The girl is spraying the plants with water*
H:*The boy is spraying the plants with water*

the transformation consisting in replacing *girl* with *boy* is just one of the possible transformation sequences that can be applied to transform T into H. In fact, other sequences would be equally possible, like the sequence of atomic operations that would remove the whole content of T and add the whole content of H.

4.1 Calculating the transformations

Several approaches to RTE operate directly at the token level, generally after applying some preprocessing, such as part-of-speech (PoS) tagging, but without computing more elaborate syntactic or semantic representations. Another approach, like the one adopted in the present work, is to work at the syntax level by using dependency grammar parsers (Kubler *et al.* 2009). The output of a dependency grammar parser is a tree whose nodes are the words in the sentence and whose labeled edges correspond to the syntactic dependencies between words. Figure 1 shows the dependency trees for the Text (T) and Hypothesis (H) introduced above.

Each node in the tree consists of the word itself, the lemmatized word with its PoS tag, and the syntactic dependency relation (dprel) with its parent node. We use MaltParser (Nivre, Hall and Nilsson 2006) trained on parts 2–21 of the Wall Street Journal section of the Penn Treebank to produce the dependency trees, while tree edit distance (see Section 5.1) is applied on the resulting trees to obtain the transformations needed to convert one tree into another. The tree edit distance problem


prep
nsubj
dobj
det
aux
det
pobj
The girl is spraying the plants with water
1 2 3 4 5 6 7 8
the girl be spray the plant with water
DT NN VBZ VBG DT NNS IN NN
prep
nsubj
dobj
det
aux
det
pobj
The boy is spraying the plants with water
1 2 3 4 5 6 7 8
the boy be spray the plant with water
DT NN VBZ VBG DT NNS IN NN
Fig. 1. Dependency tree of the Text: *The girl is spraying the plants with water* (on the left). Dependency tree of the Hypothesis: *The boy is spraying the plants with water* (on the right).

has a recursive solution that decomposes the trees into subtrees and subforests. The direct implementation of this recursive solution has exponential complexity. To limit time and space complexity, we adopt the implementation described in the paper by Zhang and Shasha (1989) which uses dynamic programming to achieve polynomial runtime. Regardless of the implementation used, one of the main characteristics of tree edit distance is that it computes the minimum number of transformations among all the possible sequences that would achieve the same result. In this context, three different types of transformation can be defined:

- Inserting: insert a node N from the tree of H into the tree of T. When N is inserted, it is attached with the dprel of the source label.
- Deleting: delete a node N from the tree of T. When N is deleted, all its children are attached to the parent of N. It is not required to explicitly delete the children of N as they are going to be either deleted or substituted on a following step.
- Replacing: change the label of a node N1 in the source tree (the tree of T) into a label of a node N2 of the target tree (the tree of H).

In addition to these, there is a fourth transformation, called *Matching*, that does not produce any changes on the nodes (i.e., it is simply applied when two nodes are equal), and which generally is not considered in the edit distance calculation. In our approach, however, the transformations are used as features for the T–H pairs classification, and the *Matching* transformation, which measures the overlap of the nodes, might be considered complementary to the other transformations and useful in estimating the similarity between the dependency trees of the fragments of text. This concept will be further developed in Section 4.3.

In our implementation, the transformations above are defined at the level of single nodes of the dependency tree (we can insert, remove and replace one node at a time while transformations consisting in inserting, removing and replacing whole subtrees are not allowed) and nodes are matched by considering both their lemma and dprel with their parent node, i.e., two nodes are considered equivalent if and only if they have exactly the same lemma and dprel.

4.2 Transformation form

The transformations are used by the classifier as features and one could envision different ways of representing them. At least two elements need to be specified: the transformation type (e.g., inserting) and the node (or nodes in the case of replacing) involved in the transformation itself. For the nodes, we need to set the level of specificity/genericity; for example, the nodes could be expressed by the lemma of the words (e.g., girl), or by combining the lemma with the PoS (e.g., girl-NN) or combining the lemma and dprel (e.g., girl-nsubj). To avoid overfitting, we adopt a representation in which the nodes in the transformation are only represented by their dependency labels, i.e.,

- *Replacing(T:node-dprel, H:node-dprel),*
- *Inserting(H:node-dprel),*
- *Deleting(T:node-dprel).*

For our running example, this would be: *Replacing(T:nsubj, H:nsubj)*, meaning that a word in T has to be replaced with another word in H and that both these words have a syntactic relation of *nsubj* with their respective parents.

4.3 The role of the knowledge resources

The use of resources like WordNet is crucial for recognizing cases where T and H use different textual expressions that preserve the entailment (e.g., *child* versus *boy*). The Knowledge Resources that we use (see Section 5.2) contain both semantic relations that preserve the entailment relation (e.g., WordNet synonyms and hypernyms) and that change the entailment relation (e.g., WordNet antonyms). With tree edit distance, we use the preserving relations in the following way:

WHEN in the Knowledge Resources there exists an entailment-preserving relation between two words, THEN those words are matched as if they were the same word.

As an example, consider the two words *toy* and *object* in the pair T:*Two dogs are running and carrying a toy in their mouths*, H:*Two dogs are running and carrying an object in their mouths*. For these two words, the following matching transformation would be produced: *Matching(T:dobj, H:dobj)*[5]. On the other hand, the non-preserving relations are used for labeling those transformations that, when applied on two words, do not preserve the entailment (e.g., white versus black, leaving versus arriving):

WHEN in the Knowledge Resources there exists a non-entailment-preserving relation between two words in a transformation, THEN we label that transformation with the label non-preserving entailment.

As a result, the transformation for our running example, the replacement of *girl* with *boy*, which are antonyms in WordNet, can be represented as *Replacing(T:nsubj, H:nsubj, non-preserving entailment).*

[5] The transformation is produced in addition to the other matching transformations (e.g., Matching(T:nsubj, H:nsubj)) calculated for those words that T and H have in common (e.g., dogs).

Table 3. *Transformations for some T–H pairs in SICK with similar meaning. On the left, the pairs with the rules applied to produce them. On the right, the transformations produced by our method. An index on T and H highlights the words in the transformation, e.g., 3 (Ins(H_3:auxpass)) means that the transformation regards the word at position 3 in H (is)*

Pairs	Transformations
Label: ENTAILMENT Rule: turn active sentences into passive T: A man is driving a car H: The car is being driven by a man	1. Rep(T_2:nsubj,H_2:nsubjpass) 2. Rep(T_1:det,H_1:det) 3. Ins(H_3:auxpass) 4. Ins(H_6:prep) 5. Rep(T_6:dobj,H_8:pobj)
Label: ENTAILMENT Rule: turn passive sentences into active T: The potato is being peeled by a woman H: One woman is peeling the potato	1. Rep(T_2:nsubjpass,H_2:nsubj) 2. Rep(T_1:det,H_1:det) 3. Del(T_4:auxpass) 4. Del(T_6:prep) 5. Rep(T_8:pobj,H_6:dobj) 6. Rep(T_7:det,H_5:det)
Label: ENTAILMENT Rule: replace words with synonyms T: A young boy is jumping into water H: A young kid is jumping into water	1. Match(T_3:nsubj,H_3:nsubj) *There exists a preserving relation between boy and kid and they are matched as if they were the same word.*
Label: ENTAILMENT Rule: add modifiers T: A wild deer is jumping a fence H: A deer is jumping a fence	1. Del(T_2:amod)
Label: ENTAILMENT Rule: replace quantifiers T: A surfer is riding a big wave H: The surfer is riding a big wave	1. Rep(T_1:det,H_1:det)

4.4 Transformations produced from SICK

As described in Section 3, the annotators have created T–H pairs with similar meaning, contrasting meaning and pairs where the truth value of one fragment cannot be inferred from the other fragment. Table 3 shows the transformations that our method produces for some examples of T–H pairs with similar meaning, while Table 4 shows the transformations for examples of the two remaining cases.

5 System architecture

We recast RTE as a classification problem in which the provided T–H pairs are classified as positive or negative examples. Figure 2 shows the system's architecture.

The T–H pairs are first preprocessed (Section 5.1) to produce their dependency trees, then a set of features (i.e., the transformations) are extracted from these trees (Section 5.3). Knowledge Resources (Section 5.2) can be used in this phase to match words with the same or similar meaning and those with opposite meanings. Finally,

Table 4. *Transformations for some T–H pairs in SICK with contrasting meaning or where the truth of one of its fragments cannot be inferred from the other one. On the left, the pairs with the rules applied to produce them. On the right, the transformations produced by our method*

Pairs	Transformations
Label: CONTRADICTION Rule: insert a negation T: The boy is playing the piano H: The boy is not playing the piano	1. Ins(T_4:neg)
Label: CONTRADICTION Rule: change determiners with opposites T: There is no dog walking along a snowdrift H: A dog is walking along a snowdrift	1. Rep(T_2:root,H_4:root) 2. Ins(H_2:nsubj) 3. Rep(T_1:expl,H_1:det) 4. Del(T_4:nsubj) 5. Rep(T_3:neg,H_3:aux) 6. Del(T_5:partmod)
Label: NEUTRAL Rule: replace words with semantic opposites T: The girl is spraying the plants with water H: The boy is spraying the plants with water	1. Rep(T_2:nsubj,H_2:nsubj,npe) where *npe* means that the transformation does not preserve the entailment.

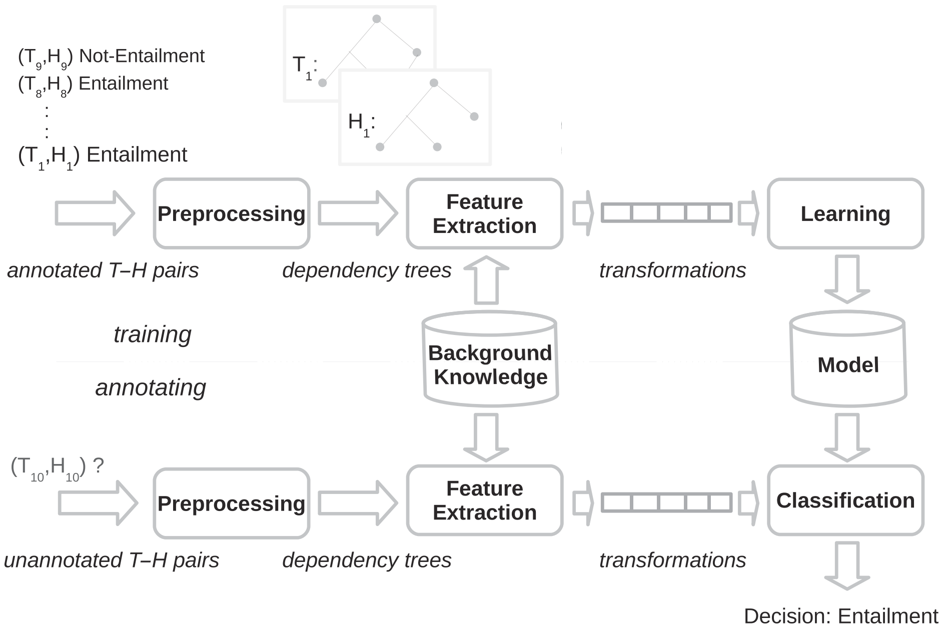


Fig. 2. System architecture.

a classifier uses the features produced (Section 5.4) to classify the T–H pairs to be annotated.

The implementation of the system was developed within the EOP (Padó *et al.* 2013; Magnini *et al.* 2014) for RTE, which is a generic architecture and a comprehensive implementation for textual inference in multiple languages that promotes the reuse of software components. The platform includes state-of-the-art algorithms for RTE relations, a large number of knowledge resources and facilities for experimenting and testing innovative approaches.

5.1 Preprocessing

The T–H pairs are preprocessed with PoS tagging, lemmatization and dependency parsing. The output of this phase consists of the dependency trees created for each pair (i.e., one tree for the Text (T) and another for the Hypothesis (H)). We use TreeTagger (Schmid 1994) for tokenization, lemmatization and PoS tagging, while MaltParser (Nivre *et al.* 2006) is used for dependency parsing. As an example, consider the pair: T:*The girl is spraying the plants with water* and H:*The boy is spraying the plants with water* for which the preprocessing output has been reported in Figure 1. This will be used as the running example in this section.

5.2 Knowledge resources

Knowledge resources are crucial in recognizing cases where T and H use different textual expressions that are semantically related and can be relevant in identifying the entailment relation (e.g., *kid* versus *boy*, *home* versus *house*). In this context, it must be noted that the component we are using for querying the Lexical Knowledge currently has some limitations. First of all, we cannot access the resources by a combination of both lemma and PoS, but by lemma only. As a consequence, it could happen, for example, that a word used as a noun in a certain context and another word used as a verb produce a positive match only because they share the same lemma. Second, resources like WordNet are used without any word sense disambiguation, i.e., a word in the text fragments could be matched with its wrong sense in the resource. In the rest of this section, we briefly report on the three resources used, namely WordNet, CatVar, and VerbOcean, and describe how their semantic relations (e.g., synonym) can be grouped into relations that preserve the entailment and that do not preserve the entailment.

WordNet (Miller 1995) is a lexical database that groups words into sets of synonyms called synsets, provides short, general definitions and records the various semantic relations between these synonym sets. From the available relations, we use the following ones: hypernym (e.g., leave → move), entailment (e.g., leave → go), synonym (e.g., leave → go away) and antonym (e.g., leave → arrive). Of these, the *hypernym*, *entailment* and *synonym* are considered to be preserving entailment, and *antonym* is considered to be non-preserving.

CatVar (Habash and Dorr 2003) is composed of uninflected words (lemmas) and their categorial (i.e., PoS) variants. For example, the words hunger (V), hunger (N), hungry (AJ) and hungriness (N) are different English variants of the same underlying concept describing the state of being hungry. The *local-entailment* relation present in CatVar is considered to be an entailment-preserving relation.

VerbOcean (Chklovski and Pantel 2004) is similar to WordNet, but applies only to verbs. VerbOcean contains lexical–semantic relations that are unique to verbs, such as stronger than (e.g., whisper → talk). Many of these relations indicate entailment at the lexical-level, and are useful for covering inference relations between verbs that do not exist in WordNet. The relations used from VerbOcean are *similar*, which preserves entailment, and *opposite of*, which is a non-preserving relation.

5.3 *Feature extraction*

Feature extraction is used in order to convert each input pair into a feature set. We represent each T–H pair as a set of binary features which are the transformations obtained by applying tree edit distance on the dependency trees produced in the previous phases. To limit time and space complexity, we use the implementation described in the paper by Zhang and Shasha (1989), as said in Section 4.1.

5.4 *Classification*

The availability of annotated data sets makes it possible to formulate the problem of RTE in terms of a classification task. For classification, we used Weka (Hall *et al.* 2009), which is a collection of machine learning algorithms for data mining tasks. Using Weka, we decided to try different algorithms, like Naive Bayes (Rish 2001), logistic regression (le Cessie and van Houwelingen 1992), J48 (i.e., an open source Java implementation of the C4.5 algorithm (Quinlan 1986) in the Weka data mining tool), Random Forests (Breiman 2001) and SVM (Vapnik 1995).

6 Experiments and results

We evaluated our approach by testing it on both the SICK and EXCITEMENT data sets. As regards SICK, the experiments have been conducted in the same way as the ones performed for SemEval-2014 Task#1. This means that we split the development set into two parts: a dev-train for training the system (3,290 examples) and a dev-test (1,644) for finding the best system configuration to be used for the test set. Achieving the best system configuration required a number of experiments: Section 6.1 describes the baseline and two basic system configurations that were used as reference point to compare other tested configurations. After choosing the best of these configurations, Section 6.2 refers to the experiments conducted using different algorithms for the classification of the T–H pairs, while Section 6.3 refers to the contribution of the knowledge resources to the final system performance. Section 6.4 compares the different approaches to node matching.

With respect to the EXCITEMENT data set, each of its four data sets comes in two parts: a dev-train that we use for training the system and a dev-test for tuning. Given that the test data set was not available at the time we were writing, in this section, we report the results we have obtained on the dev-test. We would like to highlight that the same experiments conducted on SICK were also performed on this data set to find the most promising system configurations. For reasons of space, and considering that the fine-grain annotation of SICK allows for much more refined conclusions, and that the EXCITEMENT test set will be released at a later

Table 5. *Accuracy measure for the baseline and the basic system configurations. Computing time in seconds for training and for test (in parentheses)*

Data set	Accuracy	Time(s)
Basic-system#1 (SVM)	80.4	33.9 (0.1)
Basic-system#2 (SVM w/o resources)	77.8	33.8 (0.1)
Baseline	62.1	

time, for EXCITEMENT, we show only the results for a number of configurations and this is done in Section 6.5.

Finally, Section 6.6 concludes by comparing our results on SICK with those obtained by the participants of SemEval-2014 Task#1, while for EXCITEMENT, the results are compared with those of the systems distributed with the EOP. For both data sets, we run the experiments on the same hardware infrastructure.[6]

6.1 Baseline and evaluation measures

To measure the performance of each of the system configurations that were tested, we created a baseline calculated by annotating all the T–H pairs in the dev-test with the most common entailment relation in the dev-train (i.e., neutral). Then, we explored two different basic configurations: basic-system#1 was calculated by using SVM and a linear kernel for classification. SVM is one of the most popular classification algorithms used in machine learning, while the linear kernel is faster than other kernel functions and easier to optimize. WordNet, CatVar and VerbOcean were used as resources as they are well-known, freely-available resources. Next, basic-system#2 was obtained in the same way as basic-system#1, but without using any external resources.

In Table 5, the overall system performance over the different category labels is measured in terms of accuracy (i.e., the percentage of predictions that are correct), while, in the rest of the section, the classification effectiveness over the single category is calculated in terms of the classic information retrieval notions of Precision (Pr) and Recall (Re), together with the F_1 measure.

Supervised learning methods need annotated data in order to generate efficient models. However, annotated data is relatively scarce and can be expensive to obtain. The learning curve plotted for our method (Figure 3) shows the accuracy on the dev-test as a function of the number of examples in the dev-train. Acceptable results can be obtained using only twenty per cent of the training data (980 examples). Moreover, exceeding fifty per cent of the training examples (2,467 examples) does not produce any significant improvements; it is the case that the new annotated examples cannot solve the most problematic pairs, which will be examined in the next part of this section.

[6] PC Intel(R) Xeon(R) 2 GHz, RAM 8 GB, Linux Red Hat Enterprise

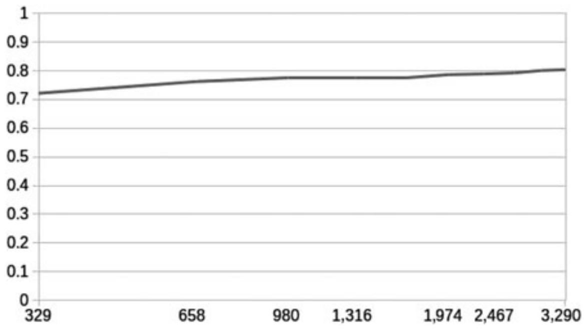


Fig. 3. Learning curve (semi-log scale) calculated with basic-system#1 (SVM).

Table 6. *Values in brackets highlight improvements or deterioration in performance with respect to values obtained by basic-system#1 (SVM). Computing time in seconds for training and for test (in parentheses)*

	Accuracy	t-test	Time(s)
SVM (2-poly kernel)	81.2 (+0.8)	($\gg$)	111.7 (5.5)
Random Forests	80.5 (+0.1)	($\gg$)	53.9 (0.9)
J48	77.7 (−2.7)	($\ll$)	25.6 (0.1)
Logistic	77.2 (−3.2)	($\ll$)	84.4 (0.2)
Naive Bayes	72.6 (−7.8)	($\ll$)	1.0 (2.9)

6.2 Algorithms evaluation

Starting from basic-system#1 (SVM), we replaced SVM with other algorithms for classification: Random Forests, J48, Logistic and Naive Bayes; we have chosen these algorithms because they are frequently reported in the literature as having good classification performance. A second degree polynomial kernel for SVM was used in order to try to capture the complex interactions that occur among the syntactic transformations. Table 6 reports on the results obtained when these algorithms are trained on the SICK dev-train and evaluated on the dev-test. The annotation ($\gg$), ($\ll$) or ($\sim$) in the t-test field of the table indicates that a specific result is statistically better ($\gg$), worse ($\ll$) or equivalent ($\sim$) compared to the results of basic-system#1 (SVM) at the significance level specified (currently 0.05).

According to this test, SVM with polynomial kernel and Random Forests performed better than SVM using linear kernel, while Naive Bayes performed poorly. These results could also be explained by considering the presence of interactions among the features extracted: if each of the transformations makes an independent contribution to the output, then algorithms based on linear functions (e.g., logistic, SVM with linear kernel, Naive Bayes) generally perform well. However, if there are complex interactions among features, as is the case here, then algorithms such as SVM with polynomial kernels of higher degree, decision trees or ensemble methods using decision trees, like Random Forests, work better, because they are specifically designed to discover these interactions.

Table 7. *One resource removed at a time*

	Accuracy	*t*-test	Coverage
-WordNet	77.5 (−2.9)	($\ll$)	0.02018
-CatVar	80.2 (−0.2)	($\ll$)	0.00047
-VerbOcean	80.7 (+0.3)	($\sim$)	0.00217

6.3 Knowledge resources evaluation

To measure the impact of the knowledge resources on recognizing the entailment relations, we started with basic-system#1 (SVM), using all the resources described in Section 5.2 and then removed one lexical resource at a time. This is reported in Table 7. In the table, the coverage of the resources is measured as the number of times we find a relation in a given resource, divided by the number of accesses to the resource.

WordNet resulted as the most important resource (−2.9 when it is not used), CatVar (−0.2) gives a small contribution and finally VerbOcean (+0.3) seems to not be useful. The good result of using WordNet can also be explained by considering its coverage, which is higher than that of the other resources. Using WordNet enables, for example, the identification the T–H pairs that had been created by replacing words with synonyms or by replacing words with semantic opposites. Concerning VerbOcean, but this is also true for all the other resources, we must point out once again that the EOP component that we are currently using for querying the Lexical Knowledge does not allow access to the resources by a combination of both lemma and PoS, but by lemma only. As a direct consequence of this, it could happen, for example, that, even though VerbOcean contains semantic relations only between verbs, words used as a noun in a certain context, and those that are actually verbs, might produce a positive match, which is used in the tree edit distance calculation. As an example of this problem, consider the pair T:*A little girl is hitting a baseball off a tee*, H:*A little girl in a pink shirt is playing t-ball and taking a swing*, where the verb *hit* in the Text (T) and the noun *swing* in the Hypothesis (H) create a positive match. The same problem can occur when accessing WordNet; for the pair T:*A monkey is pulling a dog's tail*, H:*A monkey is teasing a dog at the zoo*, the noun *tail* in T is matched with the noun *dog* in H, because they are synonyms in WordNet but with the meaning of ‘to follow’ (i.e., a verb).

6.4 Node matching evaluation

Tree edit distance computes the transformations working on the nodes in the trees and different transformations can be produced under different types of node matching. In basic-system#1 (SVM), two nodes are considered equivalent if and only if they have the exact same lemma and dprel. However, other strategies for comparing nodes are possible. Using only the lemma results in a significant drop in accuracy (−20.1); in this case, for example, a lemma that is the subject of a Text (T), can be successfully matched by tree edit distance against the same lemma playing

Table 8. *System accuracy for all three entailment relations. In parentheses, the number of examples in the data set*

	Accuracy	Pr	Re	F_1
Contradiction (1,424)	73.2	88.8	73.2	80.2
Entailment (2,821)	71.2	80.6	71.2	75.6
Neutral (5,595)	90.4	81.9	90.4	86.0

Table 9. *Confusion matrix on the SICK test set*

a	b	c	← classified as
2,523	206	61	a = Neutral
400	999	5	b = Entailment
156	35	521	c = Contradiction

another syntactic role in a Hypothesis (H) (e.g., the object). On the other hand, matching the nodes by only considering the syntactic relations lets tree edit distance match nodes that have the same syntactic role in the text fragments (e.g., the subject) but which, in fact, represent different words (e.g., man versus car, woman versus potato); this may also explain the loss in accuracy (−12.3).

6.5 System evaluation

In the previous sections, we have reported a large number of experiments in order to find the best system configuration on the dev-test. When using that configuration to train our system on the whole development set and then testing it on the test data set, we obtained 82.4 per cent accuracy. Table 8 gives details about the system's performance for all the three entailment relations of SICK i.e., contradiction, entailment and neutral. Table 9 reports the confusion matrix obtained on the SICK data set.

The confusion matrix reveals that in some cases the entailment and contradiction relations can be confused with the neutral one. An explanation for this lies in the higher presence in the data set of sentences with subjects that contain indefinite articles, where the entailment judgments are subject to human interpretation. In fact, in the data set, there are pairs annotated with neutral or contradiction labels depending on whether the subjects of their T and H were believed to refer to the same entity or two distinct ones.

The results, grouped into categories corresponding to the different expansion rules that had been applied, can be found in Table 10. To obtain S2, the annotators applied meaning-preserving rules, while to obtain S3, they used rules creating contradictory or contrasting meaning. Finally, to get sentences S4, a set of word scrambling rules were applied while ensuring that the resulting sentence was still meaningful. The results in the table are only reported for those rules that generate at least ten T–H

Table 10. *System accuracy on the T–H pairs generated by applying the preserving, negative and scrambling rules*

				Accuracy	F_1
preserving					
Add modifiers	S1	S2	48	92.6	96.2
	S2	S1	238	97.5	98.7
Turn adjectives into relative clauses	S1	S2	91	84.4	91.6
	S2	S1	98	87.8	93.5
Turn compounds into relative clauses	S1	S2	28	78.6	88.0
	S2	S1	28	90.3	94.9
Replace words with synonyms	S1	S2	481	77.4	82.6
	S2	S1	365	68.6	73.5
Turn active sentences into passive	S1	S2	146	82.4	83.4
	S2	S1	135	87.3	93.2
Replace quantifiers	S2	S1	134	96.8	98.4
Expand agentive nouns	S2	S1	133	83.6	89.8
negative					
Insert a negation	S1	S3	204	95.2	96.3
	S3	S1	215	97.2	97.4
Change determiners with opposites	S1	S3	307	89.5	94.0
	S3	S1	301	87.6	91.7
Replace words with semantic opposites	S1	S3	468	72.5	78.5
	S3	S1	465	70.0	77.3
others					
Scramble words	S1	S4	187	80.9	83.5
	S4	S1	190	82.2	81.7

pairs in the test data set, while sentences S1, S2, S3 and S4 can appear in T and H or vice versa. The annotation of the pairs was done using the primary configuration.

Errors in the system's output regard the pairs that have been created by replacing words with synonyms and with semantic opposites. Here, a major issue can be the lack of coverage of the knowledge base used for word matching; as an example of words for which we do not have any match in the knowledge base, consider *grass* and *lawn* in the pair T:*A brown and white dog is playing on the grass*, H:*A brown and white dog is playing on the lawn*. Other examples refer to words used with opposite meaning, like *playing* and *sleeping* in T:*Children in red shirts are playing in the leaves*, H:*Children in red shirts are sleeping in the leaves*; or the use of partial synonyms like *marsh* and *river* in T:*A monkey is wading through a marsh*, H:*A monkey is wading through a river*. One instance where one could expect to have better performance is that of the pairs created by turning compounds into relative clauses. Examples of such pairs are T:*A dog is pushing a toddler into a rain puddle*, H:*A dog is pushing a toddler into a puddle of rain*, or pairs like T:*A person is looking at a bike designed for motocross that is lying on its side and another is racing by*, H:*A person is looking at a motocross bike that is lying on its side and another is racing by*. Even though for the first pair, our approach creates simple transformations (i.e., replacing the noun compound *rain* in T with the nominal modifier *rain* in H; and

Table 11. *Results of different configurations for each data set of EXCITEMENT. Precision, Recall and F1 measure are calculated on the positive class*

	Accuracy	Pr	Re	F_1	Configuration
Mix					
Balanced	78.7	76.9	81.3	79.0	SVM (2-poly kernel)
	77.7	76.0	80.2	78.0	RandomForest
	75.4	73.6	78.3	75.9	basic-system#1 (SVM)
	75.3	73.4	78.4	75.8	basic-system#2 (SVM w/o res)
	74.7	74.4	74.4	74.4	J48
	68.1	66.5	71.2	68.8	NaiveBayes
	50.7	–	–	–	baseline
Unbalanced	85.3	77.8	59.4	67.4	Random Forests
	84.7	79.7	53.7	64.2	SVM (2-poly kernel)
	81.6	66.7	55.6	60.6	J48
	81.3	66.4	53.8	59.4	basic-system#1 (SVM)
	80.7	65.3	51.5	57.6	basic-system#2 (SVM w/o res)
	74.5	–	–	–	baseline
	72.8	47.2	58.1	52.1	NaiveBayes
Pure					
Balanced	66.4	65.1	70.6	67.7	Naive Bayes
	66.1	64.8	70.3	67.5	SVM (2-poly kernel)
	65.9	63.0	76.9	69.3	basic-system#1 (SVM)
	66.0	62.8	78.7	69.8	basic-system#2 (SVM w/o res)
	64.4	64.0	65.7	64.8	RandomForest
	63.1	62.0	68.0	64.8	J48
	50.0	–	–	–	baseline
Unbalanced	87.3	–	–	–	baseline
	77.4	19.2	24.5	21.5	RandomForest
	76.9	19.2	25.5	21.9	SVM (2-poly kernel)
	75.0	18.2	27.6	21.9	basic-system#1 (SVM)
	74.9	18.0	27.3	21.7	basic-system#2 (SVM w/o res)
	73.7	16.0	25.1	19.5	J48
	70.2	19.7	43.7	27.1	Naive Bayes

then inserting a new node for the case relation *of*), the second pair produces a much larger number of transformations, which our system finds hard to process. Another factor contributing to this result is the limited number of annotated examples for this rule.

Regarding EXCITEMENT, we have conducted several experiments on its four configurations. Table 11 reports the results we obtained. In the table, basic-system#1 (SVM) and basic-system#2 (SVM w/o resources) are calculated in the same way as for the SICK experiments. The baseline was calculated by annotating all the T–H pairs with the most common entailment relation in the data sets (i.e., non-entailment).

For this set of experiments, an important point to highlight is that the lexical resources we used had almost no effect. As indicated by comparing basic-system#1 (SVM) (which uses the lexical resources) and basic-system#2 (SVM w/o resources). The reason for this is that their coverage is close to zero and more specific domain-based resources would be necessary. Another issue that comes to light when working on the four splits of the data set is that the performance levels of the classifiers are relatively close to each other, meaning that the transformations can be successfully exploited by most of the classifiers. When we consider the mix-balanced and mix-unbalanced data sets, SVM and Random Forests algorithms perform as well as they did for SICK. SVM, which is known to be robust to overfitting, also obtained good performance on the pure-balanced data set. Finally, working with the pure-unbalanced data set is a demanding task for all the classifiers because the syntactic transformations extracted from the training data set might not accurately represent the data in the test set. Here, a simple classifier like Naive Bayes, not so finely tuned to contingent characteristics of the data set, performs a bit better than the others. It should be noted that obtaining high accuracy values on this split is trivial, since a trivial rejector, i.e., the classifier that trivially assigns all pairs to the most heavily populated category (i.e., non-entailment) as baseline does, can indeed achieve very high accuracy, even though there are no applications in which one would be interested in such a classifier. In this case, F_1 values can offer additional insight. However, accuracy values are not the only factor to consider when selecting a classifier. For example, for real-time applications, one could prefer higher values of annotation speed, and might be willing to compromise on accuracy in order to achieve that; in this case, SVM with linear kernel and Random Forests would be preferable to SVM with second-degree polynomial kernel.

6.6 A comparison with other approaches

This section provides a comparison of the results obtained by AdArte with the ones obtained by different methods. For SICK, this is done by comparing AdArte with the Illinois-LH system (Lai and Hockenmaier 2014) and the other systems at SemEval-2014 Task#1 (see Table 12). In addition, we compared AdArte with systems that, when working on data set such as RTE-3, obtained state-of-the-art results: TIE is a system which combines the outcomes of several scoring functions into a classifier, while P1EDA uses various independent aligners to compute any evidence for or against entailment. Given that the current implementation of these systems does not support multi-class classification (that is required to annotate SICK), we evaluated them on two-class entailment problem. In this case, all the examples in SICK annotated with contradiction and neutral labels were converted to non-entailment.

AdArte's results are close to those of UNAL-NLP, which was ranked the first among the systems that did not use compositional models. Purely compositional models have lower performance; haLF obtained 69.4 per cent of accuracy, while the system presented by Illinois, which used a combination of the two models, was ranked first (84.6 per cent). The results of TIE (Wang and Neumann 2007) and P1EDA (Noh, *et al.* 2015) can be explained by considering that these approaches might not work well with the phenomena present in the data set, such as negations, given that their alignment-based methods are not able to capture them. The approach by Bowman *et al.* (2015) shows that the representations learned by a neural network

Table 12. *AdArte as compared with the SemEval-2014 Task#1 systems and the Bowman's neural network model. Below, AdArte, TIE and P1EDA are evaluated on two-class entailment problem*

Team Id	Accuracy
Illinois-LH-run1	84.575
ECNU-run1	83.641
UNAL-NLP-run1	83.053
AdArte	**82.4**
SemantiKLUE-run1	82.322
The-Meaning-Factory-run1	81.591
Bowman-LSTM	80.8
CECL-ALL-run1	79.988
BUAP-run1	79.663
UoW-run1	78.526
UEdinburgh-run1	77.106
UIO-Lien-run1	77.004
FBK-TR-run3	75.401
StanfordNLP-run5	74.488
UTexas-run1	73.229
Yamraj-run1	70.753
asjai-run5	69.758
haLF-run2	69.413
RTM-DCU-run1	67.201
UANLPCourse-run2	48.731
AdArte	**86.3**
TIE	78.8
P1EDA	77.6

model on a large high-quality corpus such as the Stanford Natural Language Inference corpus can be used to improve performance on a standard challenge data set: 80.8 of accuracy versus 71.3 that is the accuracy when training on SICK alone.

Concerning the EXCITEMENT data sets, our results are compared with those of the other systems implemented within the EOP (see Table 13) (i.e., they were made available in the EOP GitHub repository[7]), and with SemantiKLUE (Proisl *et al.* 2014), which was one of the top five systems at SemEval-2014 Task#1, and which has also been tested on this data set.

From the table, it is apparent that in the case of the two data sets built by evenly distributing the T–H pairs of different topics in the dev-train and dev-test (i.e., mix data sets) our method is better than others at learning patterns and regularities from data. In contrast, on the data sets where the examples in the training are not representative of the examples in test (i.e., pure data sets), the difference between

[7] `https://github.com/hltfbk/Excitement-Open-Platform/wiki/Development-test-suites-and-reports/3615810d0d24eef46ef34843cad10898dac19f09`

Table 13. *Systems ranked by Accuracy values. A number after the systems shows their position in the ranking when the F_1 for the positive class is considered*

		Accuracy	F_1	Systems
Mix	Balanced	**78.7**	**79.0**	**AdArte**[1]
		76.5	74.5	SemantiKLUE[2]
		67.0	64.8	P1EDA[6]
		64.1	72.3	BIUTEE[3]
		61.7	65.2	TIE-run2[5]
		61.7	64.6	EditDistance[7]
		57.4	65.6	TIE-run1[4]
	Unbalanced	**85.3**	**67.4**	**AdArte**[1]
		83.9	61.3	SemantiKLUE[3]
		79.1	62.7	BIUTEE[2]
		68.4	53.7	EditDistance[4]
		62.1	30.6	TIE-run-2[6]
		48.3	32.4	TIE-run-1[5]
Pure	Balanced	66.7	69.2	P1EDA[2]
		66.4	67.7	**AdArte**[4]
		65.7	64.0	SemantiKLUE[6]
		62.9	69.1	TIE[3]
		62.1	71.3	BIUTEE[1]
		55.7	66.6	EditDistance[5]
	Unbalanced	83.4	29.2	SemantiKLUE[3]
		80.4	39.2	BIUTEE[1]
		77.4	**21.5**	**AdArte**[5]
		63.2	33.4	EditDistance[2]
		59.5	21.4	TIE-run-2[6]
		38.3	24.4	TIE-run-1[4]

our method and the others becomes smaller. The difference is null in the case of the pure-balanced data set, whereas on the pure-unbalanced data, systems can have good accuracy on the majority class (i.e., non-entailment) but very poor accuracy on the minority class (i.e., entailment) due to the influence that the majority class has on the training criteria. In fact, many algorithms for classification are accuracy-driven. In this sense, a system like EditDistance, but whose simple classifier can be configured to maximize the F_1 measure during training, produces, on the pure-unbalanced data set, F_1 values that are above average compared to the other systems. Good results have been obtained on all the splits by one of the top five systems at SemEval-2014 Task#1, SemantiKLUE (Proisl *et al.* 2014), which, in addition, proved to be portable to new domains. Finally, it is worth highlighting that the lack of capitalization and punctuation marks information in the data set, and the presence of imperatives and elliptical phrases as well, does not seem to have affected our results with respect to other approaches based on shallow linguistic analysis, for example, the one used to implement edit distance models.

7 Conclusions and outlook

In this paper, we have described an approach to RTE based on tree-level transformations. We have seen how the T–H pairs can be represented by their dependency trees and how the transformations can be calculated from these trees by tree edit distance. The transformations have then been used as features in a classifier to label the pairs as positive or negative. Lastly, the implemented AdArte system has been made publicly available through the EOP, a generic architecture and a comprehensive implementation for textual inference in multiple languages. Unlike other existing systems, AdArte is designed to work with larger data sets. The system produces thousands of transformations and, consequently, requires that a certain number of annotated examples be available to learn a model. In this context, we have evaluated AdArte on two new data sets that present this feature: SICK, used at SemEval-2014 Task#1, and EXCITEMENT, a new data set of email feedbacks.

With respect to SICK, a comparison of our results with those of SemEval-2014 systems reveals promising results: the transformations are able to capture the main syntactic and lexical phenomena in the data set and they can be successfully used with a variety of classifiers. This result is all the more interesting if we consider that some systems at SemEval-2014 use approaches that would be difficult to adapt to new domains such as the EXCITEMENT one (e.g., the denotational features used in the Illinois system are fairly strongly tied to the SICK domain) or more complex machine learning techniques, such as co-training used in ECNU to exploit large amounts of unlabeled data, which are out of the scope of this work.

With regards to the EXCITEMENT data set, we have compared our method with the systems in the EOP. In the case of the two data sets built by evenly distributing the T–H pairs referring to different topics in the dev-train and dev-test (i.e., mix data sets), our method appears better than others at learning patterns and regularities from data. In contrast, on the data sets where the examples in the training are not so representative of those in the test set (i.e., the pure data sets), the difference between our method and the best systems in the EOP becomes smaller or void, revealing a certain degree of overfitting. This phenomenon is even more visible when working with pure-unbalanced data sets, where edit distance models, which learn simple threshold values, outperform other systems.

Despite the satisfactory results, our method has two main types of limitations: limitations regarding the size of the data set and regarding the use of the knowledge resources. The first type of limitation is due to the approach itself, which is suitable for data sets which consist of a sufficient number of examples to learn a model. Possible examples of data sets for which our approach is not indicated could be the RTE-3 data sets which consist of 800 annotated pairs. When tested on this data set, AdArte produces lower accuracy values (58.3 per cent) than other methods tested on the same data set. In fact on the Third PASCAL RTE Challenge (RTE-3), participants reported accuracy values ranging from forty-nine per cent to eighty per cent, while, for example, another system such as P1EDA obtained 67.0 per cent.

Another limitation of our approach stems from cases that cannot be solved by comparing the syntactic structures of the T–H pairs. This is due to the fact that syntax alone is not sufficient to determine entailment in more complex cases that require some degree of text understanding. As an example, consider the following T–H pair, where the existing entailment relation between T:*Carl Smith collided with a concrete lamp-post while skating and suffered a skull fracture that caused a coma. When he failed to regain consciousness, his parents on August 8 consented to his life support machine being turned off*, and H:*Carl Smith died on August 8* cannot be determined by merely considering the syntactic differences between them. Another limitation of this type is the inability to deal with phrasal verbs in the current implementation of the system. One example of such phrasal verbs would be: *I am counting on you to make dinner while I am out*. In our approach, tree edit distance can only work at the level of single nodes and, as a consequence, it cannot match phrases such as *count on* with *trust* or *go on* with *continue*. Missing these matches prevents us from using resources like WordNet, which would be able to identify that *go on* and *continue* are synonyms and they should match. We are going to address this issue by providing an additional preprocessing phase able to take as input the dependency trees and then collapse the nodes of a phrasal verb into a single node. The last issue refers to the component that we are using for querying the lexical knowledge: it cannot access the resources by a combination of both lemma and PoS, but by lemma only, and, moreover, resources like WordNet are used without any word sense disambiguation. We will also address this issue by using a new software component that has been made available in order to overcome these limitations.

We plan to evaluate AdArte in languages other than English, such as German and Italian; for this purpose, we will use the OMQ and EXCITEMENT Italian data sets[8] created from manually categorized German and Italian customer requests, and that were recently added into the EOP for evaluation.

Finally, considering the promising results, we obtained on the two data sets tested, we are going to use AdArte in a real-world application developed within the EXCITEMENT project. More precisely, this application aims to provide an easy-to-use interface for analysts to gain insight from a mass of customer interactions through TE. The application takes as input the customer interactions from various service channels of a typical contact center (e.g., transcribed calls and emails). Then, an automatic text fragment extraction is applied on the customer interactions to extract meaningful text fragments such as reasons for dissatisfaction or other customer events that could be interesting for analysis. AdArte will get the set of relevant fragments as input to produce a structured hierarchy based on the entailment relations that hold between these text fragments, which will then be displayed by the application.

References

Bos, J. and Markert, K. 2005. Recognising textual entailment with logical inference. In *EMNLP-05*, Stroudsburg, PA, USA, Association for Computational Linguistics, pp. 628–35.

[8] `https://github.com/hltfbk/EOP-1.2.1/wiki/Data-Sets`

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. 2015. Learning natural language inference from a large annotated corpus. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Stroudsburg, PA. Association for Computational Linguistics, pp. 632–42.

Breiman, L. 2001. Random forests. *Machine Learning* **45**(1): 5–32.

Chklovski, T., and Pantel, P. 2004. VerbOcean: mining the web for fine-grained semantic verb relations. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A Meeting of SIGDAT, a Special Interest Group of the ACL, Held in Conjunction with ACL 2004, 25–26 July 2004,* Barcelona, Spain, pp. 33–40.

Dagan, I., Dolan, B., Magnini, B., and Roth, D. 2009. Recognizing textual entailment: rational, evaluation and approaches. *Journal of Natural Language Engineering* **15**(4): i–xvii.

Dagan, I., Glickman, O., and Magnini, B. 2005. The PASCAL recognising textual entailment challenge. In *Proceedings of the 1st PASCAL Challenges Workshop on Recognising Textual Entailment*, Southampton, UK.

Dagan, I., Roth, D., Sammons, M., and Massimo Zanzotto, F. 2013. *Recognizing Textual Entailment: Models and Applications*. Synthesis Lectures on Human Language Technologies. San Rafael, CA, USA, Morgan & Claypool Publishers.

Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, B. 2007. The third PASCAL recognising textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, Prague, Czech Republic.

Habash, N., and Dorr, B. 2003. A categorial variation database for English. In *NAACL/HLT 2003, Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, Stroudsburg, PA, USA, Association for Computational Linguistics, pp. 96–102.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. 2009. The Weka data mining software: an update. *SIGKDD Explorations Newsletter* **11**(1): 10–18, November.

Harmeling, S. 2009. Inferring textual entailment with a probabilistically sound calculus. *Journal of Natural Language Engineering* **15**(4): 459–477.

Hastie, T., Tibshirani, R., and Friedman, J. 2001. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc.

Heilman, M., and 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 Association for Computational Linguistics*, HLT '10, Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 1011–19.

Kotlerman, L., Dagan, I., Magnini, B., and Bentivogli, L. 2015. Textual entailment graphs. *Journal of Natural Language Engineering Special Issue on Graphs for NLP* **21**(5): 699–724.

Kouylekov, M., and Magnini, B. 2005. Recognizing textual entailment with tree edit distance algorithms. In *PASCAL Challenges on RTE*, Berlin, Heidelberg, Springer-Verlag, pp. 17–20.

Kubler, S., McDonald, R., Nivre, J., and Hirst, G. 2009. *Dependency Parsing*. San Rafael, CA, USA, Morgan and Claypool Publishers.

Lai, A., and Hockenmaier, J. 2014. Illinois-LH: a denotational and distributional approach to semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University, pp. 329–34.

le Cessie, S., and van Houwelingen, J. C. 1992. Ridge estimators in logistic regression. *Applied Statistics* **41**(1): 191–201.

MacCartney, B. 2007. Natural logic for textual inference. In *ACL Workshop on Textual Entailment and Paraphrasing*, Stroudsburg, PA, USA, Association for Computational Linguistics.

Magnini, B., Zanoli, R., Dagan, I., Eichler, K., Neumann, G., Noh, T.-G., Padó, S., Stern, A., and Levy, O. 2014. The excitement open platform for textual inferences. In *Proceedings of* *the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22–27, 2014,* Baltimore, MD, USA, System Demonstrations, pp. 43–8.

Marelli, M., Bentivogli, L., Baroni, M., Bernardi, R., Menini, S., and Zamparelli, R. 2014a. SemEval-2014 Task1: evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University, pp. 1–8.

Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., and Zamparelli, R. 2014b. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014),* Reykjavik, Iceland, May 26–31, 2014, pp. 216–23.

Miller, G. A. 1995. WordNet: a lexical database for English. *Communications of the ACM* **38**(11): 39–41, November.

Nivre, J., Hall, J., and Nilsson, J. 2006. MaltParser: a data-driven parser-generator for dependency parsing. In *Proceedings of LREC-2006*, Paris, France, European Language Resource Association, pp. 2216–19.

Noh, T.-G., Padó, S., Shwartz, V., Dagan, I., Nastase, V., Eichler, K., Kotlerman, L., and Adler, M. 2015. Multi-level alignments as an extensible representation basis for textual entailment algorithms. In *Proceedings of *SEM 2015*, Stroudsburg, PA, USA, Association for Computational Linguistics.

Padó, S., Noh, G., Stern, A., Wang, R., and Zanoli, R. 2013. Design and realization of a modular architecture for textual entailment. *Journal of Natural Language Engineering* **1**(12): 1–34.

Proisl, T., Evert, S., Greiner, P., and Kabashi, B. 2014. Semantiklue: robust semantic similarity at multiple levels using maximum weight matching. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University, pp. 532–40.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning* **1**(1): 81–106.

Rish, I. 2001. An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, pp. 41–46.

Schmid, H. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, Manchester, UK, pp. 44–9.

Stern, A., and Dagan, I. 2012. BIUTEE: a modular open-source system for recognizing textual entailment. In *Proceedings of the ACL 2012 System Demonstrations*, ACL'12, Stroudsburg, PA, USA. Association for Computational Linguistics, pp. 73–8.

Vapnik, V. N. 1995. *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag New York Inc.

Wang, R., and Neumann, G. 2007. Recognizing textual entailment using a subsequence kernel method. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence, July 22–26, 2007,* Vancouver, British Columbia, Canada, pp. 937–43.

Zhang, K., and Shasha, D. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing* **18**(6): 1245–62, December.