An Extensible Probabilistic Transformation-based Approach to the Third Recognizing Textual Entailment Challenge

Stefan Harmeling
Institute of Adaptive and Neural Computation
School of Informatics, Edinburgh University, Scotland
stefan.harmeling@ed.ac.uk

Abstract

We introduce a system for textual entailment that is based on a probabilistic model of entailment. The model is defined using some calculus of transformations on dependency trees, which is characterized by the fact that derivations in that calculus preserve the truth only with a certain probability. We also describe a possible set of transformations (and with it implicitly a calculus) that was successfully applied to the RTE3 challenge data. However, our system can be improved in many ways and we see it as the starting point for a promising new approach to textual entailment.

1 Introduction

Textual entailment recognition asks the question whether a piece of text like

> The Cassini Spacecraft has taken images from July 22, 2006 that show rivers and lakes present on Saturn's moon Titan.

implies a hypothesis like

> The Cassini Spacecraft reached Titan.

There exists already many interesting approaches to this problem, see (Dagan et al., 2005; Bar-Haim et al., 2006) for various recent efforts and our paper wont try to fully reinvent the wheel. Instead it will present some work in progress that tries to model the probability of entailment in terms of ideas motivated by approaches like the edit-distance (Kouylekov and Magnini, 2005; Kouylekov and Magnini, 2006; Tatu et al., 2006; Adams, 2006). However, instead of defining some distance based on edits, we will generate derivations in some calculus that is able to transform dependency parse trees. The special property of our calculus is that the truth is only preserved with a certain probability along its derivations. This might sound like a disadvantage. However, in commonsense reasoning there is usual a lot of uncertainty due the fact that it is impossible to formalize *all* world knowledge. We think that probabilities might help us in such situations where it is impossible to include everything into the model, but in which nonetheless we want to do reasoning.

2 Main idea

First of all, let us assume that the text and the hypothesis of an textual entailment example are represented as dependency trees T and H. We would like to formalize the probability that T entails H with some model $p_\theta(T \models H)$ parametrized by a vector θ. In order to define $p_\theta(T \models H)$ we first introduce the probability of preserving truth along syntactic derivations in some calculus $T \vdash_\tau H$ which we informally introduce next.

Suppose we are given n transformations $\mathsf{TF}_1, \dots, \mathsf{TF}_n$ that are designed to modify dependency trees. For each such transformation TF_j, the probability of preserving truth is modelled as a constant value θ_j independent of the dependency tree T it is applied to, i.e.

$$p_\theta(T \vdash_{\mathsf{TF}_j} \mathsf{TF}_j(T)) = \theta_j \quad \text{for all } T, \tag{1}$$

with parameter θ being the vector of all θ_j. The idea is that applying a transformation to T could also create a dependency tree that is sometimes not entailed by T anymore. Consider e.g. the transformation that extracts an appositive and adds a new sentence for it. Usually this is correct, but there are situations in which the appositive appears inside a quote, where it might lead to a wrong conclusion. Thus it makes sense to consider probabilities to deal with imperfect calculi.

We call an n-tuple of such transformations a derivation, which we denote by τ with $\ell(\tau) = n$. Let τ_j count the number of times TF_j appears in τ. Furthermore, let $\tau(T)$ be the result of applying the transformations in τ to some dependency tree T, e.g. for $\tau = (\mathsf{TF}_3, \mathsf{TF}_3, \mathsf{TF}_{17})$ with $\ell(\tau) = 3$ we have $\tau(T) = \mathsf{TF}_{17}(\mathsf{TF}_3(\mathsf{TF}_3(T)))$.

Suppose that a derivation $\tau = (t_1, \ldots, t_{\ell(\tau)})$ derives H from T, i.e. $\tau(T) = H$. Then we define the probability of preserving the truth along the derivation τ as the product of the preservation probabilities of the transformations involved[1]:

$$p_\theta(T \vdash_\tau H) = \prod_{i=1}^{\ell(\tau)-1} p_\theta(T_i \vdash_{t_i} T_{i+1}) = \prod_{j=1}^{n} \theta_j^{\tau_j} \quad (2)$$

with $T_1 = T$, $T_{i+1} = t_i(T_i)$ and $T_{\ell(\tau)} = H$. Note that even though for a certain dependency tree T applying different derivations τ and σ can result in the same tree, i.e. $\tau(T) = \sigma(T)$, their probabilities of preserving truth can be different, since the probabilities depend only the transformations applied.

In the previous paragraphs we have defined probabilities of preserving truth for all finite length derivations in the calculus. This allows us now to define the probability of $T \models H$ to be the maximal probability over all possible derivations,

$$p_\theta(T \models H) = \max_{\tau:\tau(T)=H} p_\theta(T \vdash_\tau H) = \max_{\tau:\tau(T)=H} \prod_{j=1}^{n} \theta_j^{\tau_j}. \quad (3)$$

In the following we introduce a set of transformations that is able to transform any text into any hypothesis and we will propose a heuristic that generates such derivations.

[1]Note, that this definition is similar to the idea of "transitive chaining" introduced in (Dagan and Glickman, 2004).

3 Details

3.1 Preprocessing and parsing

For preprocessing we apply the following steps to the text string and the hypothesis string:

(1) Remove white space, dots and quotations marks at beginning and end.
(2) Remove trailing points of abbreviations.
(3) Remove space between names, e.g. 'Pat Smith' becomes 'PatSmith'.
(4) Unify numbers, e.g. resolve 'million'.
(5) Unify dates, e.g. '5 June' becomes 'June 5'.

We then split the text string into sentences simply by splitting at all locations containing a dot followed by a space. The resulting strings are fed to the Stanford Parser (de Marneffe et al., 2006; Klein and Manning, 2003) with its included pretrained model and options '-retainTmpSubcategories' and '-splitTMP 1'. This allows us to generate dependency trees the nodes of which contain a single stemmed word, its part-of-speech tag and its dependency tag (as produced using the parser's output options 'wordsAndTags' and 'typedDependencies', see (de Marneffe et al., 2006)). For the stemming we apply the function 'morphy' of the NLTK-LITE toolbox (Bird, 2005). If the text string contains more than one sentence, they will be combined to a single dependency tree with a common root node. Let us from now on refer to the dependency trees of text and hypothesis by T and H.

3.2 Generating derivations

The heuristic described in the following generates a derivation that transforms T into H. For brevity we will use in the text the abbreviations of the transformations as listed in Tab. 1.

(1) Resolve appositives and relative clauses. All derivations for some T and H start by converting any existing appositives and relative clauses in T to new sentences that are added to T. For each sentence that was added in this step, the applied transformation, ATS or RTS, is appended to τ.

(2) Calculate how H can clamp to T. Often there are several possibilities to assign all or some of the words in H to words in T. For simplicity our system currently ignores certain grammatical parts which

SS	substitute synonym
SN	substitute number
SNE	substitute named entity
SI	substitute identity
SHE	substitute hypernym
SHO	substitute hyponym
SC	substitute currency
SP	substitute pronoun
GTC	grammar tag change
CP	change prep
SA	substitute antonym
DOS	del other sents
RUP	remove unclamped parts
RUN	remove unclamped negs
RUNO	remove unclamped negs oddity
MCU	move clamped up
RRN	restructure remove noun
RAN	restructure add noun
RRV	restructure remove verb
RAV	restructure add verb
RPD	restructure pos depth
RND	restructure neg depth
RHNC	restructure h neg count
RHNO	restructure h neg oddity
ATP	active to passive
PTA	passive to active
ATS	appos to sent
RTS	rcmod to sent

Table 1: Current transformations with their abbr.

are auxiliaries ('aux'), determiners ('det'), prepositions ('prep') and possessives ('poss'). Furthermore, we currently ignore words of those parts of speech (POS) that are not verbs ('VB'), nouns ('NN'), adjectives ('JJ'), adverbs ('RB'), pronouns ('PR'), cardinals ('CD') or dollar signs ('$'). For all other words w_H in H and words w_T in T we calculate whether w_T can be substituted by w_H. For this we employ amongst simple heuristics also WordNet 2.1 (Fellbaum, 1998) as described next:

(1) Are the tokens and POS tags of w_T and w_H identical? If yes, return (1, *'identity'*).
(2) If the POS tags of w_T and w_H indicate that both words appear in WordNet continue with (3) otherwise with (8).
(3) Are they antonyms in WordNet? If yes, return (2, *'antonym'*).
(4) Are they synonyms in WordNet? If yes, return (2, *'synonym'*).
(5) Does w_H appear in the hypernym hierarchy of w_T in WordNet? If yes, return (z, *'hyponym'*) with z being the distance, i.e. w_T is a hyponym of w_H.
(6) Does w_T appear in the hypernym hierarchy of w_H in WordNet? If yes, return (z, *'hypernym'*) with z being the distance, i.e. w_T is a hypernym of w_H
(7) Are they named entities that share certain parts of their strings? If yes, return (z, *'named entity'*) with z being larger dependent on how different they are.
(8) Is w_T a pronoun and w_H a noun? If yes, return (2, *'pronoun'*).
(9) Are w_T and w_H exactly matching cardinals? If yes, return (1, *'number'*).
(10) Are w_T and w_H identical currencies? If yes, return (1, *'currency'*).
(11) Are w_T and w_H both currencies? If yes, return (2, *'currency'*).

Note that along the hierarchy in WordNet we also look one step along the "derived form" pointer to allow a noun like 'winner' be substitutable by the verb 'win'. If a word w_T is substitutable by a word w_H, we say that w_T and w_H are *clamped*. We call the whole assignment that assigns some or all words of H to words in T a *clamp*. Since usually a single word w_H is clamped to several words in T, we will often have several different clamps. E.g. if H has three words each of which is clamped to four words in T there are sixty-four possible clamps in total, i.e. sixty-four possible ways to clamp the words in H to words in T.

Each of these different clamps gives rise to a different derivation. However, let us for simplicity continue to focus on a single clamp and see how to complete a single derivation τ.

(3) Substitute the clamped words. If w_H and w_T are clamped, we know what their relationship is: e.g. (3, *hypernym*) means that we have to go three steps up w_H's hypernym-hierarchy in WordNet to reach w_T. Thus we have to apply three times the transformation SHE to substitute w_T by w_H, which we reflect in τ by appending three times SHE to it. Similarly, we add other transformations for other relations. The substitution of w_T with w_H might also trigger other transformations, such as PTA, ATP, CP and GTC which try to adjust the surrounding grammatical structure. All applied transformations will be appended to the derivation τ.

(4) Pick the sentence with the most clamps. After substituting all clamped words, we simply pick the sentence in T with the most clamped words and delete the others using DOS. E.g. if T consists of three sentences, after this step T will only contain the single sentence with the most clamps and DOS will be appended twice to τ.

(5) Remove subtrees that do not contain clamped nodes. After this step we add for each removed node the transformation RUP to τ. Then we add RUN for each removed negation modifier ('neg') and additionally RUNO if the number of removed negation is odd. RUNO is a somewhat artificial transformation and acts more like a flag. This might be changed in future sets of transformation to better comply with the transformation metaphor.

(6) Move the clamped nodes closer to the root and remove unclamped subtree. Again we do some counting before and after this step, which determines the transformations to add to τ. In particular we count how many verbs are passed by moving clamped nodes up. For each passed verb we add MCU to τ.

(7) Restructure and add the missing pieces. The definition in Eq. (3) requires that any T can be transformed into any H, otherwise the maximum is undefined. In the last step we will thus remove all words in T which are not needed for H and add all missing words to T and restructure until T becomes H. For the bookkeeping we count the number of nouns, verbs and negation modifier that have to be added and removed. Furthermore, we count how many levels up or down we need to move words in T such that they match the structure in H. For all these countings we add accordingly as many transformations RRN, RRV, RAN, RAV, RPD, RND, RHNC, RHNO (see Tab. 1 for short explanations).

Finally, the completed derivation τ with $\tau(T) = H$ is converted to a 28-dimensional feature vector $[\tau_1, \ldots, \tau_{28}]^\top$ using the notion of τ_j which has been defined in Sec. 2.

3.3 Estimating the parameters

Let $D_{tr} = \{(T_1, H_1, y_1), \ldots, (T_{800}, H_{800}, y_{800})\}$ be the training examples with $y_i \in \{0, 1\}$ indicating entailment. For brevity we define

$$f_i(\theta) = p_\theta(T_i \models H_i) \tag{4}$$

to abbreviate the probability of entailment modelled as outline in Sec. 2. Then the data likelihood can be written as:

$$p_\theta(D_{tr}) = \prod_{i=1}^{800} f_i(\theta)^{y_i} (1 - f_i(\theta))^{(1-y_i)} \tag{5}$$

We would like to maximize $p_\theta(D_{tr})$ in term of the vector θ. However, the maxima in Eq. (3) make this optimization difficult. For the submission to the RTE3 challenge we choose the following way to approximate it:

(1) Generate for each example pair several derivations (as described in the previous section) and choose the eight shortest ones. If there are less than eight derivations available, copy the shortest ones to end up with eight (some of which could be identical).
(2) There are now $8 \cdot 800$ derivations in total. We denote the corresponding feature vectors by $x_1, \ldots, x_{6400}$. Note that x_i is a vector containing the countings of the different transformations. E.g. if the corresponding derivation was τ, then $x_{ij} = \tau_j$.
(3) Similarly copy the training labels y_i to match those 6400 feature vectors, i.e. now our data becomes $D_{tr} = \{(x_1, y_1), \ldots, (x_{6400}, y_{6400})\}$.
(4) Replacing $f_i(\theta)$ by

$$g_i(\theta) = \prod_j \theta_j^{x_{ij}} \tag{6}$$

the data likelihood becomes:

$$p_\theta(D_{tr}) = \prod_{i=1}^{6400} g_i(\theta)^{y_i} (1 - g_i(\theta))^{(1-y_i)} \tag{7}$$

(5) Replace furthermore each θ_j by

$$\sigma(z_j) = \frac{1}{1 + \exp(-z_j)} \tag{8}$$

with σ being the sigmoidal function, which ensures that the values for θ_j stay between zero and one.
(6) Maximize $p_z(D_{tr})$ in terms of $z = [z_1, \ldots, z_n]^\top$ using gradient ascent.
(7) Calculate $\theta_j = \sigma(z_j)$ for all j.

3.4 Classifying the test data

Having estimated the parameter vector θ we can apply the trained model to the test data D_{te} to infer its unknown labels. Since we only generate some derivations and can not try all possible—as would be required by Eq. (3)—we again transform the test data into 6400 feature vectors $x_1, \dots, x_{6400}$. Note that $x_1, \dots, x_8$ are the feature vectors belonging to the first test example (T_1, H_1), and so forth. To approximate the probability of entailment we take the maximum over the eight feature vectors assigned to each test example, e.g. for (T_1, H_1),

$$p_\theta(T_1 \models H_1) \approx \max_{i \in \{1,\dots,8\}} \prod_{j=1}^{28} \theta_j^{x_{ij}} \tag{9}$$

and analogously for the other test examples. The class label and herewith the answer to the question whether T_i entails H_i is obtained be checking whether $p_\theta(T_i \models H_i)$ is above a certain threshold, which we can determine using the training set. This completes the description of the system behind our first run of our RTE3 submission.

3.5 Logistic Regression

The second run of our RTE3 submission is motivated by the following observation: introducing a weight vector with entries $w_j = \log \theta_j$ and using logarithms, we can rewrite Eq. (3) as

$$\log p_\theta(T \models H) = \max_{\tau:\tau(T)=H} \sum_{j=1}^{n} \tau_j w_j. \tag{10}$$

The probability of entailment becomes the maximum of several linear expressions with the additional constraint $w_j < 0$ for all j which ensures that θ_j is a probability. In order to compare with another linear classifier we applied as the second run logistic regression to the data. Again we used eight derivations/feature vectors per training example to estimate the parameters of the logistic regression. Also with the test data we applied the weight vector to eight derivations/feature vectors per test example and choose the largest result which was then threshold to obtain a label.

4 Results

The first fact we see from the official RTE3 results in Tab. 3 is that our system is better than random. However, with 56% and 57.75% it is not much better. From the task specific data we see that it completely failed on the information extraction (IE) and the summarization (SUM) data. On the other hand it has reached good results well above 60% for the information retrieval (IR) and question answering (QA) data. From the accuracies of the training data in Tab. 3 we see that there was some overfitting.

We also applied our system to the RTE2 challenge data. The results are shown in Tab. 2 and show that our system is not yet competitive with last year's best systems. It is curious that in the RTE2 data the SUM task appears simpler than the other tasks while in this year's data IR and QA seem to be the easiest.

RTE 2	overall	IE	IR	QA	SUM
AccTr	0.5950	0.5700	0.5850	0.5500	0.6750
AccTe	0.5675	0.5000	0.5850	0.5600	0.6250
AccTr	0.6050	0.5700	0.5550	0.5800	0.7150
AccTe	0.5725	0.5000	0.5800	0.5800	0.6300

Table 2: Results for the RTE2 data. Shown are the accuracies on the training and test sets. First two lines for the first run (transformation-based model) and the next two lines for the second run (logistic regression).

RTE 3	overall	IE	IR	QA	SUM
AccTr	0.6475	0.5750	0.6350	0.7600	0.6200
AccTe	0.5600	0.4700	0.6250	0.6450	0.5000
PreTe	0.5813	0.5162	0.6214	0.6881	0.5607
AccTr	0.6550	0.5600	0.6300	0.7850	0.6450
AccTe	0.5775	0.5000	0.6300	0.6700	0.5100
PreTe	0.5952	0.5562	0.6172	0.7003	0.5693

Table 3: Official results on the RTE3 test data and inofficial results on the corresponding training data. Shown are the accuracies and average precision on the test data. First three lines for the first run (transformation-based model) and the next three lines for the second run (logistic regression).

5 Future work and conclusion

As already mentioned, this paper presents work in progress and we hope to improve our system in the near future. For the RTE3 challenge our main goal was to get a system running before the deadline. However, we had to make a lot of compromises/simplifications to achieve that goal.

Even though our current results suggest that right now our system might not be able to compete with the better systems from last year's challenge we see the potential that our architecture provides a useful platform on which one can test and evolve different sets of transformations on dependency trees. Again, we note that such a set of transformations can be seen as a calculus that preserves truth only with a certain probability, which is an interesting concept to follow up on. Furthermore, this idea of a probabilistic calculus is not limited to dependency trees but could equally well applied to other representations of text.

Besides working on more powerful and faithful transformations, our system might be improved also simply by replacing our ad hoc solutions for the preprocessing and sentence-splitting. We should also try different parsers and see how they compare for our purposes. Since our approach is based on a probabilistic model, we could also try to incorporate several optional parse trees (as a probabilistic parser might be able to create) with their respective probabilities and create a system that uses probabilities in a consistent way all the way from tagging/parsing to inferring entailment.

Acknowledgement

The author is grateful for valuable discussions with Chris Williams and Amos Storkey. This research was supported by the EU-PASCAL network of excellence (IST- 2002-506778) and through a European Community Marie Curie Fellowship (MEIF-CT-2005-025578). Furthermore, the author is thankful to the organizers of the RTE challenges and to the creators of WordNet, NLTK-LITE and the Stanford Parser for sharing their software with the scientific community.

References

R. Adams. 2006. Textual entailment through extended lexical overlap. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

R. Bar-Haim, I. Dagan, B. Dolan, L. Ferro, D. Giampiccolo, B. Magnini, and I. Szpektor, editors. 2006. *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

S. Bird. 2005. NLTK-Lite: Efficient scripting for natural language processing. In *4th International Conference on Natural Language Processing*, pages 1–8.

I. Dagan and O. Glickman. 2004. Probabilistic textual entailment: Generic applied modeling of language variability. In *PASCAL Workshop on Learning Methods for Text Understanding and Mining, Grenoble*.

I. Dagan, O. Glickman, and B. Magnini, editors. 2005. *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

M.-C. de Marneffe, B. MacCartney, and C.D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *International Conference on Language Resources and Evaluation (LREC)*.

C. Fellbaum. 1998. *WordNet: an electronic lexical database*. MIT Press.

D. Klein and C.D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*.

M. Kouylekov and B. Magnini. 2005. Recognizing textual entailment with tree edit distance algorithms. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

M. Kouylekov and B. Magnini. 2006. Tree edit distance for recognizing textual entailment: Estimating the cost of insertion. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

M. Tatu, B. Iles, J. Slavick, A. Novischi, and D. Moldovan. 2006. COGEX at the second recognizing textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.