A Survey of Text Entailment until June 2012

Arindam Bhattacharya

June 26, 2012

Abstract

Recognizing Textual Entailment is a task that recognizes pairs of natural language expressions, such that a human who reads (and trusts) the first element of a pair would most likely infer that the other element is also true. Textual Entailment is useful in a wide range of natural language processing applications, including question answering, summarization, text generation, and machine translation. In this document we summarize the key ideas and approaches from the area, current up to June 2012, also pointing to prominent articles and resources.

1 Introduction

Textual Entailment (TE) is a directional relation between text fragments. The relation holds whenever the truth of one text fragment follows from another text. The entailing piece of text is termed *text* and the entailed piece is called the *hypothesis*. As truly noted by [4], Textual entailment bears similarity with the famous Turing Test. to determine whether a machine can have the ability to think, as accessing knowledge sources and drawing inference are the primary ingredient for an intelligent system. As instance of entailment is illustrated by Example 1.1.

Example 1.1:

T: *Vodafone is giving free 3G trial with its new GSM SIMs.*

H: *Vodafone provides GSM services.*

1.1 Classification of Various Textual Entailment Algorithms

Textual Entailment algorithms are mainly classified based on the approach used represent the text and the hypothesis, to make it suitable for consumption by the algorithm. The preprocessing phase of the algorithm takes the natural language text and hypothesis and produces suitable representations for them.

2 Textual Entailment Based on Lexical Methods

Several paraphrase recognition methods operate directly on the input surface strings, possibly after applying some pre-processing, such as part-of-speech (POS) tagging or named-entity recognition, but without computing more elaborate syntactic or semantic representations. They make the entailment decision solely based on the lexical evidences [5].

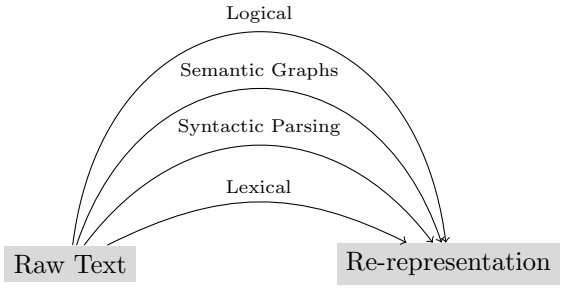

Figure 1: Various Representations

2.1 Preprocessing

In lexical approaches, preprocessing step involves *tokenization*, *stemming/lemmatization* and identifying the *stop words*. Stop words *e.g. the, a etc.*, unlike *content words*, does not contribute to recognition of entailment. This is because they occur too frequently to imply any entailment. Certain systems also carry out some deeper pre-processing tasks such as:

- **Phrasal Verb Recognition:** This step identifies all the phrasal verbs in both text and hypothesis. Examples of phrasal verbs are *take off, put on etc.*
- **Idiom processing:** Idiom is an expression, word, or phrase that has a figurative meaning that is comprehended in regard to a common use of that expression that is separate from the literal meaning or definition of the words of which it is made. There are estimated to be at least 25,000 idiomatic expressions in the English language [2]. Examples of some idioms are: *A Bird In The Hand Is Worth Two In The Bush, A Picture Paints a Thousand Words etc.* Since they mean something different from what they mean, lexical approach would fail. Therefore they are required to be treated separately. In this step, known idioms are identified and are replaced by actual meaning.
- **Named Entity Recognition and Normalization:** Named entities such as name of person, company, *etc.*, are represented in various forms. This step identifies the named entities in text and hypothesis, and normalizes them to some single notation.
- **Date/Time arguments:** This step is similar to Named Entity Recognition except that it identifies date and time elements.

An example:

Example 2.2:

T: Eying the huge market potential, currently led by Google, Yahoo took over search company Overture Services Inc. last year.

H: Yahoo acquired Overture

In the example Overture Services Inc. and Overture are normalized by named entity recognition and the phrasal verb took over is mapped to acquired.

2.2 Representation

After the preprocessing, the text T and the hypothesis H are re-represented, in case of lexical approaches as one of the following:

- Bag-of-words: Both T and H are represented as a set of words.
- n-grams: Sequence of n tokens are grouped together. Bag of words is an extreme case of n-gram, with n=1.

Example: *The fixed routine of a bedtime story before sleeping has a relaxing effect.*

- Bag-of words: The, fixed, routine, of, a, bedtime, story, before, sleeping, has, relaxing, effect
- Bigram model (n-gram with n=2): The fixed, fixed routine, routine of, of a, a bedtime, bedtime story, story before, before sleeping, sleeping has, has a, a relaxing, relaxing effect

2.3 Control Strategy and Decision Making

The lexical approaches employ a *single pass* control strategy [1]. That means unlike iterative methods, they reach the decision in a single iteration. Decision making is done based on a certain threshold (decided experimentally) over the similarity scores generated by the algorithms. The similarity scores are calculated based mainly on WordNet distances.

2.4 Example: Local Lexical Matching Algorithm

Here is an algorithm for textual entailment based on lexical approach (figure 2.1).

- In preprocessing step, the algorithm just removes the stop words.
- It uses a bag of words for representation of T and H.
- It uses Wordnet to find similarities between the text and the hypothesis.
- The $LexicalCompare()$ (figure 2.2) procedure is used to find the similarity score based on various Wordnet parameters.
- It uses a single pass. The various thresholds $d_{Hyp}, d_{Mer}, d_{Mem}$ are determined for Hypernym, Meronym and MemberOf distances.
- It finally returns the matching score. This score is again subjected to a threshold for the final entailment decision.

The algorithm is overly simplistic and it is easy to find examples where it is not going to work. For example:

- **T:** Regan attended a ceremony in Washington to commemorate the landings in Normandy.
- **H:** Washington is located in Normandy.

would falsely be predicted as correct entailment by the system. But surprisingly, its found to work well with respect to current evaluation measures (LLM has the accuracy of 68.4% on RTE-3 data set).

INPUT: Text T and Hypothesis H.
OUTPUT: The matching score.
for all *word* in T and H **do**
  if *word* in $stopWordList$ **then**
    remove *word*;
  end if
  if no words left in T or H **then**
    return 0;
  end if
end for
$numberMatched = 0$;
for all *word* W_T in T **do**
  $Lemma_T$ = Lemmatize(W_T);
  for all *word* W_H in H **do**
    $Lemma_H$ = Lemmatize(W_H);
    if LexicalCompare($Lemma_H, Lemma_T$) **then**
      $numberMatched++$;
    end if
  end for
end for

Figure 2: LLM Algorithm

if $Lemma_H == Lemma_T$ **then**
  return TRUE;
end if
if HypernymDistance(W_H , W_T) $\leq d_{Hyp}$ **then**
  return TRUE;
end if
if MeronymDistance(W_H , W_T) $\leq d_{Mer}$ **then**
  return TRUE;
end if
if MemberOfDistance(W_H , W_T) $\leq d_{Mem}$ **then**
  return TRUE;
end if
if SynonymOf(W_H , W_T) **then**
  return TRUE;
end if

Figure 3: Lexical Compare Procedure

3 Methods Based on Syntactic Similarity

Bag-of-words or n-gram model representation can take us only so far. For deeper understanding of the sentences we eventually would require to show how the words in the sentence affects each other, *i.e.* how are the words dependent on each other. These dependencies could be syntactic (*e.g.* which phrase does the word belong) or semantic (*e.g.* what role does the word play). In this chapter, we explorer different methods that harnessess syntactic dependencies.

3.1 Entailment Model

Haghighi *et al.* [8] suggests an entailment model over syntactic graphs derived from Collins' head propogation rules [7]. The model is as follows. For hypothesis graph H, and text graph T, a matching M is a mapping from the vertices of H to those of T. For vertex v in H, $M(v)$ denotes its *match* in T.

As is common in statistical machine translation, nodes in H are allowed to map to fictitious $NULL$ vertices in T if necessary.

Suppose the cost of matching M is $Cost(M)$. If $\mathbf{M}$ is the set of such matchings, the cost of matching H to T is defined to be:

$$MatchCost(H,T) = \min_{M \in \mathbf{M}} Cost(M) \tag{1}$$

Let $VertexSub(v, M(v))$ be a model which gives us a cost in [0, 1], for substituting vertex v in H for $M(v)$ in T. Then,

$$VertexCost(M) = \frac{1}{Z} \sum_{v \in H_v} w(v) * VertexSub(v, M(v)) \tag{2}$$

where $w(v)$ is relative importance of vertex v, and Z is the normalizer, $\sum_{allv} w(v)$. Now, consider an edge $e = (v, v') \in H_E$, and let $\phi_M(e)$ be the path from $M(v)$ to $M(v')$ in T. Let $PathSub(e, \phi_M(e))$ be a model for assessing the *cost* of substituting a direct relation $e \in H_E$ for its counterpart, $\phi_M(e)$, under the matching. This leads to formulation of $RelationCost(M)$ in a similar fashion:

$$RelationCost(M) = \frac{1}{Z} \sum_{e \in H_E} w(e) * PathSub(e, \phi_M(e)) \tag{3}$$

The total cost function could then be expressed as a convex combination of the two cost functions as:

$$Cost(M) = \alpha * VertexCost(M) + (1 - \alpha) * RelationCost(M) \tag{4}$$

3.2 Enhancements to Dependency Graphs

Using the above dependency graph as the base, various enhancements can be applied to the graphs [11].

1. **Collapse Collocations and Named-Entities:** We *collapse* dependency nodes which represent named entities (*e.g.* nodes [Arindam] and [Bhattacharya] could be collapsed into [Arindam Bhattacharya]) and also collocations listed in WordNet, including phrasal verbs (*e.g.*, *blow off* in *He blew off his work*).

2. **Dependency Folding:** It was found that it is useful to fold certain dependencies (*e.g.* modifying prepositions such as *in*, *under etc.*) so that modifiers became labels connecting the modifier's governor and dependent directly.

3. **Semantic Role Labeling:** Graph representation was augmented with Probank-style semantic roles. Each predicate adds an arc labeled with the appropriate semantic role to the head of the argument phrase. This

helps to create links between words which share a deep semantic relation not evident in the surface syntax. Additionally, modifying phrases are labeled with their semantic types (*e.g.*, *Pakistan got independence in* $[1947]_{Temporal}$.), which should be useful in Question Answering tasks.

4. **Co-reference links:** Using a co-resolution tagger, *coref* links are added throughout the graph. These links, connecting referring and referent entities, are the only link between two sentences.

4 Approaches that Employ Machine Learning

At a certain level, entailment problem can be considered simply as a classification problem. Precisely, it is a two class problem, **YES** class and **NO** class (figure 4). Given the text T and the hypothesis H, we need to extract various similarity features, forming a feature vector. This feature vector is given to a classifier (*e.g.* SVM), which classifies it as true (YES) or false (NO) entailment.

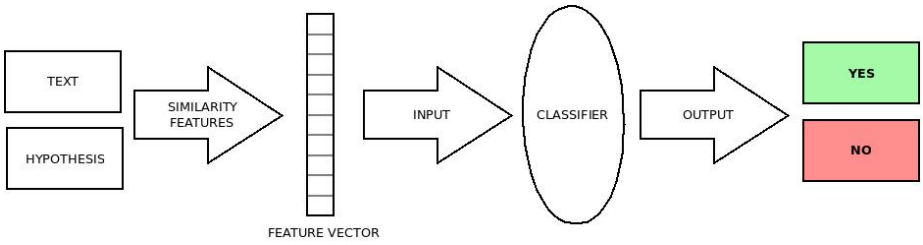

Figure 4: Textual Entailment as a Classification Task

4.1 Feature Space

In machine learning approaches, once we get a feature vector, classification task is easy. So, the main concern is finding a suitable feature space.

The various possible feature spaces are:

- **Distance Features** Features of some distance between T and H.
- **Entailment Triggers** Features that triggers entailment (or non-entailment)
- **Pair Feature** Content of T-H pair

4.1.1 Distance Features

The distance features models the distance between the text and the hypothesis in some way. These features can be of wide range, for example:

- Number of words in common
- Length of longest common subsequnce
- Longest common syntactic sub-tree

For example:
Example 4.3:

T: At the end of the year, all solid companies pay dividends.

H: At the end of the year, all solid insurance companies pay dividends.

The above example, possible $\langle feature, value \rangle$ pair could be $\langle WordsInCommon, 11 \rangle$ or $\langle LongestSubsequence, 8 \rangle$.

4.1.2 Entailment Triggers

Another possible feature space is that of *entailment triggers*. Entailment triggers are events that help us detect entailment [10]. The entailment trigger features are:

Polarity Features: Presence or absence of negative polarity.

> **Example:** Textual Entailment is very difficult. ⇒ Textual Entailment is *not* hard.

Antonymy Features: Presence or absence of antonymous words in T and H.

> **Example:** Spartans were brave soldiers. ⇒ Spartan soldiers were *coward*.

Adjunct Features: Dropping/adding of syntactic adjunct.

> **Example** John goes fishing. ⇒ John goes fishing *everyday*.

4.1.3 Pair Features

In this feature space, we try to model the content of T and H rather than modeling the distance between them. While using this feature space, choosing the right features is crucial. Following example illustrates why it could be bad if we select wrong features. Given an example in training set,
Example 4.4:

T: At the end of the year, all solid companies pay dividends.

H: At the end of the year, all solid insurance companies pay dividends.

if we choose *bag of words* as *pair features*, we would end up learning some pretty stupid rules such as

> $T \Rightarrow H$ as when T contains *end*.

This happens because there is no way to correlate the features of T and H. Does that mean that pair features are totally irrelevant? Well, with the right set of features, this space is found to give best performance [15].

4.1.4 Effectively using Pair Feature Space

Lets take an example to explain the effective use of pair feature space. Consider
Example 4.5:

T: *At the end of the year, all solid companies pay dividends.*

H_1**:** *At the end of the year, all solid insurance companies pay dividends.*

H_2**:** *At the end of the year, all solid companies pay cash dividends.*

If we would have taken distance feature, it would plot both $\langle T, H_1 =\rangle$ and $\langle T, H_2\rangle$ to be same point in feature space. What we need is something that can model the *content* and the *structure* if the T and H.

4.1.5 The Kernel Trick

To solve the above problem, we use a syntactic pair feature space [3]. To do this, instead of taking the features separately, we use kernels to represent the distance between two example pairs.

Cross Pair Similarity:

$$K(< T', H' >, < T'', H'' >) = K(< T', T'' >) + K(< H', H'' >)$$

We desire the definition of $K(P_1, P_2)$ to be such that it can exploit the *rewrite rules* of the examples. For this, placeholders were introduced in the syntactic tree. The cross pair similarity is based on the distance between syntactic trees with *co-indexed leaves*:
$K(< T', H' >, < T'', H'' >) =$

$$\max_{c \in C}(K_T(t(H', c), t(H'', i)) + K_T(t(T', c), t(T'', i))) \quad (5)$$

where,

C is the set of all correspondences between anchors of (T',H') and (T'',H'')

$t(S, c)$ renames the anchors in the parse tree S with configuration c.

i is the identity mapping

$K_T(t_1, t_2)$ is a similarity measure between t_1 and t_2

How the rewrite rules are exploited is illustrated by following example. Consider the the pair:
Example 4.6:

T: *Chapman killed Lennon.*

H: *Lennon died.*

Using the syntactic pair features and the kernel representation described above we can learn useful rules (unlike those learned using *bag of words*) such as in figure 5.

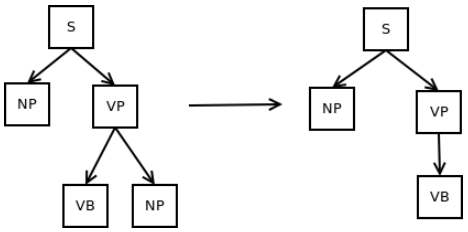

Figure 5: Exploiting rewrite rules

4.2 Classifier Algorithms

As discussed above, selection suitable features is the main problem in machine learning based approaches. Once done, any off-the-shelf classifier can be used for the classification tasks.

Using syntactic pair feature kernel and SVM, the accuracy of the system on RTE 3 data set was 62.97%. Using the approach together with lexical distance features, however, raises the accuracy up to 68.26 [15].

5 Logic Based Approach

One possibility is to map the language expressions to logical meaning representations, and then rely on logical entailment checks, possibly by invoking theorem provers. Bos and Markert, 2005 [4] and Tatu and Moldovan, 2005 [14] utilized theorem prover first order predicates. In the case of textual entailment, this involves generating pairs of formulae $< \phi(T), \phi(H)$, for **T** and **H**, and then checking if $(\phi(T)B) \models \phi(H)$, where **B** contains meaning postulates and common sense knowledge. In practice, however, it may be very difficult to formulate a reasonably complete **B**. A partial solution to this problem is to obtain common sense knowledge from resources like WordNet [12]. An example of element in **B** could be:

$$\forall x \forall y \; assassinate(x, y) \Rightarrow kill(x, y) \tag{6}$$

Additional axioms can be obtained from FrameNets frames [6], as discussed for example by Tatu *et al.* [14], or similar resources. Roughly speaking, a frame is the representation of a prototypical situation (*e.g.*, a purchase), which also identifies the situations main roles (*e.g.*, the buyer, the entity bought), the types of entities (*e.g.*, person) that can play these roles, and possibly relations (*e.g.*, causation, inheritance) to other prototypical situations (other frames). VerbNet [13] also specifies, among other information, semantic frames for English verbs. Online encyclopedias have also been used to obtain background knowledge by extracting particular types of information (*e.g.*, is-a relationships) from their articles [9]. Another approach is to use no particular **B** (meaning postulates and common sense knowledge), and measure how difficult it is to satisfy both $\phi(T)$ and $\phi(H)$, in the case of textual entailment recognition, compared to satisfying $\phi(T)$ on its own.

6 Summary

Textual entailment is currently a popular research topic. Although textual entailment can be described in terms of logical entailment, they are usually intended to capture human intuitions that may not be as strict as logical entailment; and although logic-based methods have been developed, most methods operate at the surface, syntactic, or shallow semantic level, with dependency trees being a particularly popular representation. Recognition methods, which classify input pairs of natural language expressions (or templates) as correct or incorrect paraphrases or textual entailment pairs, often rely on supervised machine learning to combine similarity measures possibly operating at different representation levels (lexical, syntactic, semantic). More recently, approaches that search for sequences of transformations that connect the two input expressions are also gaining popularity, and they exploit paraphrasing or textual entailment rules extracted from large corpora. The RTE challenges provide a significant thrust to recognition work, and they have helped establish benchmarks and attract more researchers.

References

[1] Rod Adams, Gabriel Nicolae, Cristina Nicolae, and Sanda Harabagiu. Textual entailment through extended lexical overlap and lexico-semantic matching. *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing - RTE '07*, (June):119, 2007.

[2] Stacey Bailey. Identifying linguistic knowledge for textual inference. *cllt.osu.edu*, 2004.

[3] R. Bar-Haim, Ido Dagan, I. Greental, and E. Shnarch. Semantic inference at the lexical-syntactic level. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 22, page 871. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[4] Johan Bos and Katja Markert. Recognising textual entailment with logical inference. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, number October, pages 628–635, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[5] S. Clinchant, Cyril Goutte, and Eric Gaussier. Lexical entailment for information retrieval. *Advances in Information Retrieval*, pages 217–228, 2006.

[6] Kevin Bretonnel Cohen. Natural Language Processing for Online Applications: Text Retrieval, Extraction and Categorization (review). *Language*, 80(1):178–178, 2004.

[7] Micheal Collins. Head-driven statistical models for natural language parsing. *Ph.D. thesis, University of Pennsylvania.*, 1999.

[8] Aria D. Haghighi, Andrew Y. Ng, and Christopher D. Manning. Robust textual inference via graph matching. *Proceedings of the conference on*

Human Language Technology and Empirical Methods in Natural Language Processing - HLT '05, pages 387–394, 2005.

[9] Adrian Iftene. *Textual entailment*. PhD thesis, University of Tel Aviv, 2009.

[10] Diana Inkpen, Darren Kipp, and Vivi Nastase. Machine learning experiments for textual entailment. *Proceedings of the second RTE Challenge, Venice-Italy*, 2006.

[11] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[12] George A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[13] Karin Kipper Schuler. *VERBNET: A Broad-Coverage, Comprehensive Verb Lexicon*. PhD thesis, University of Pennsylvania.

[14] Marta Tatu, Brandon Iles, John Slavick, Adrian Novischi, and Dan Moldovan. Cogex at the second recognizing textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 104–109, 2006.

[15] Fabio Massimo Zanzotto, Marco Pennacchiotti, and Alessandro Moschitti. A machine learning approach to textual entailment recognition. *Natural Language Engineering*, 15(04):551–583, September 2009.