

IT3105 Project III: Recognizing Textual Entailment

Lecture notes for Part 4

Erwin Marsi

Fall 2011

1 Introduction

In this final part of project, we look at several ways to improve your RTE system. We first discuss a number of additional information sources - lexical semantic relations, named entities, dependency relations and polarity - and how these may be stated in the form a feature. Next we discuss a number of other options for improving performance such as text normalization and co-reference resolution. The Appendix lists some NLP resources that may be exploited.

2 Features

2.1 Lexical semantic similarity

Consider the following example:

(1) T: The 26-member International Energy Agency said, Friday, that member countries would **release** oil to help relieve the U.S. fuel crisis caused by Hurricane Katrina.

⇒

H: Foreign oil reserves will be **made available** to the U.S. in the wake of Hurricane Katrina.

To understand the entailment relation, we need the information that *made available* means essentially the same thing as *release*, that is, *to make available* is a **synonym** for *release*.

Another example:

(2) T: Moog's **synthesiser**, which bears his name, revolutionised music from the 1960s onwards, and was used by bands like The Beatles and The Doors.

⇒

H Moog's **instruments** were used by The Beatles and The Doors among others.

Here we need to know that a *synthesizer* is a kind of *instrument*, which is another way of saying that *synthesizer* is a **hyponym** of *instrument* or, conversely, that *instrument* is a **hyperonym** of *synthesizer*.

The field of linguistics that studies the meaning of words, and relations between words as far as their meaning is concerned, is called **lexical semantics**. A lot of lexical semantic information about English words is encoded in Princeton's **WordNet** for English.¹ WordNet WordNet is a large lexical database of nouns, verbs, adjectives and adverbs which are grouped into sets of synonyms, or so-called **synsets**. Each synset expresses a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. WordNet is freely and publicly available for download. It is a frequently used tool for computational linguistics and natural language processing, including RTE.

As an example, here are the two different senses of the noun *synthesizer* according to WordNet 3.1, characterized by a synset of synonyms, a definition and an example sentence.

1. {synthesist, synthesizer, synthesiser} (an intellectual who synthesizes or uses synthetic methods)
2. {synthesizer, synthesiser} ((music) an electronic instrument (usually played with a keyboard) that generates and modifies sounds electronically and can imitate a variety of other musical instruments)

Looking up the direct hyperonyms of the first sense of *instrument* gives us synsets {**keyboard instrument**} and {**electronic instrument, electronic musical instrument**}. Both of these share the synset {**musical instrument**} as their hyperonym, which in turn has (one sense of) *instrument* as its hyperonym. Therefore *instrument* is an indirect hyperonym of *synthesizer*.

Wordnet can thus be regarded as a large graph with synsets as nodes and lexical semantic relations as labeled edges. In fact, some interfaces to Wordnet let you browse through this graph.² The subgraph that defines the relation between *synthesizer* and *instrument* is shown in Figure (2).

It is very helpful that we can derive from WordNet that *synthesizer* and *instrument* are related by following hyperonym relations. However, there is a problem with this approach, because by the same method we can establish that *dinosaur* and *chewing gum* are related, because if we keep following the chain of hyperonym relations we ultimately end up at a very abstract concept such as *entity* that covers almost every noun. Clearly, we need some measure

¹<http://wordnet.princeton.edu/>

²One nice example developed at NTNU is <http://wordvis.com/>

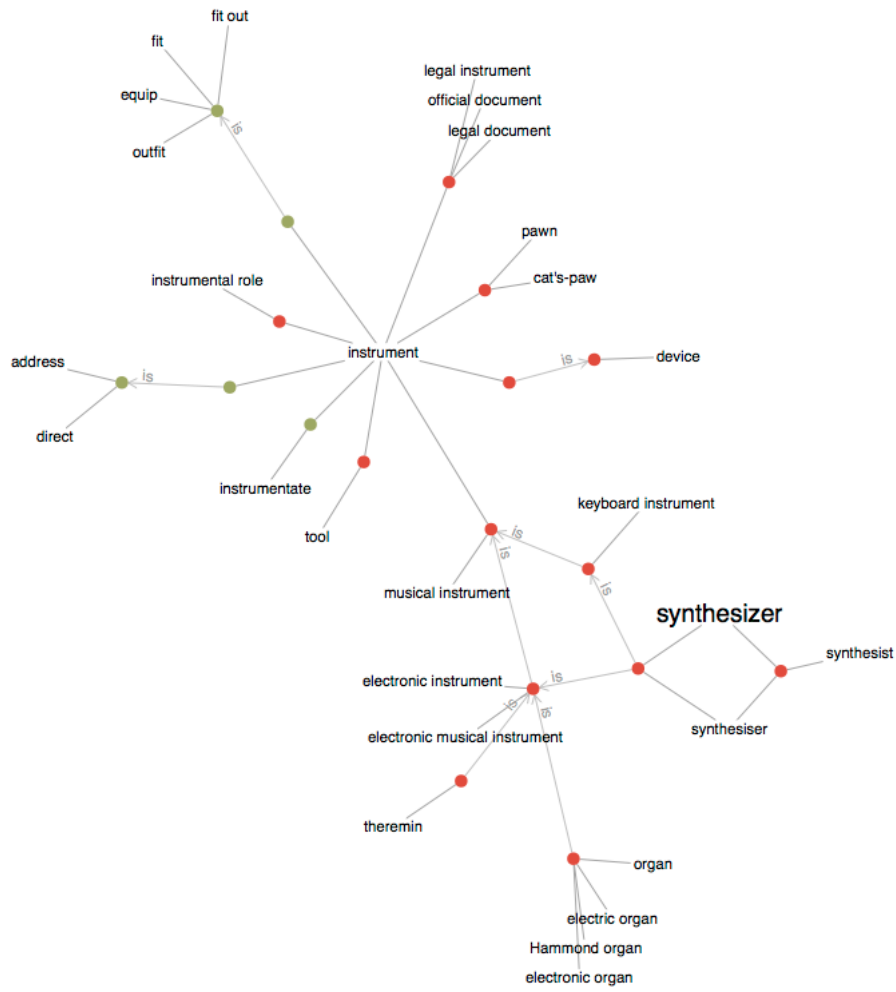


Figure 1: Visualization of a part of Wordnet as a graph, showing the relation between *synthesizer* and its indirect hyperonym *instrument*

of graph distance to express our intuition that *synthesizer* and *instrument* are much more closely related than *dinosaur* and *chewing gum*.

It turns out that the most obvious candidate – path length – works to some extent, but is far from satisfactory. The reason is that some sections of WordNet, perhaps as a result of how we categorize the world in the English language, are far more detailed than other sections. As a result, the “conceptual distance” between a word and its direct hyperonym can be relatively small in one part (as between *musical instrument* and *instrument*) and relatively large in others (as between *security blanket* and *thing*). The quest for computing semantic similarity or semantic relatedness between WordNet concepts has given rise to a range of proposed measures, many of which are implemented in toolkits for accessing WordNet. From a practical point of view, the Jiang & Conrath measure and the Lin measure appear to be good choices, even though the differences between measures are probably insignificant as far as RTE performance is concerned.

There is another practical issue that has to do with the *sense* of a word. As we saw above, for example, *synthesizer* has two senses, roughly that of a musical instrument and of an intellectual who synthesizes ideas. The latter sense is a hyperonym of *intellectual*, which is in itself a hyperonym of *person*. However, concluding that in example (2) *synthesizer* and the *Moog* are referring to the same person would clearly be a mistake. Thus in order to use WordNet properly, we need to identify its correct sense (in addition to its part of speech as noun, verb, adjective or adverb).

The task of determining the correct sense of word, provided a predefined set of senses, is called **Word Sense Disambiguation**. This is a hard problem in Natural Language Processing. Unfortunately the preprocessed RTE data does not include sense tags. There are basically two ways to handle this problem. The first one is to use some existing WSD software package to perform disambiguation; some suggestions can be found in Appendix I. The second one is to simply ignore the problem, compute similarity measures for all the senses of the words involved, and take the maximum score.

This brings us to the final question of how to compute a single feature value for the similarity between Text and Hypothesis in terms of the lexical semantic similarity between individual words. Here is one possible way to compute a measure called Lexical Semantic Similarity (LSS):

$$LSS(H, T) = \frac{\sum_{w_i \in H} \arg \max_{w_j \in T} Sim(w_i, w_j)}{\sum_{w_i \in H} Sim(w_i, w_i)} \quad (1)$$

This assumes a function *Sim* that returns the lexical similarity between two words as a real number between zero and one, where by definition $Sim(w, w) = 1$. The *argmax* operator selects the w_j for which $Sim(w_i, w_j)$ is maximal. That is, for each Hypothesis word w_i , we select the highest lexical similarity with any Text word w_j .

This equation may be extended in a number of ways. For example, instead of words, use word plus part-of-speech combinations, or even word plus part-of-speech plus sense number, to prevent spurious matches. Another possibility is to weight each term, for example, by the IDF weight of each Hypothesis word. Smart feature engineering has a big impact on performance, often far larger than the choice of learning algorithm or parameter optimization.

2.2 Named entities

Named-Entity Recognition is the process of tagging elements in text into predefined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc. Below is an example of a sentence pair in which named entities are tagged with online version of the Illinois Named Entity Tagger³, which recognizes names of persons (PER), locations (LOC), organizations (ORG) and miscellaneous names (MISC).

(3) T: The 26-member [*ORG* International Energy Agency] said, Friday, that member countries would release oil to help relieve the [*LOC* U.S.] fuel crisis caused by [*MISC* Hurricane Katrina] .

⇒

H: Foreign oil reserves will be made available to the [*LOC* U.S.] in the wake of [*MISC* Hurricane Katrina] .

The idea is that all named entities in H must also occur in T, otherwise it is unlikely that an entailment relation holds. It is straight forward to come up with a (normalized) measure that expresses the overlap in named entities. Although there are obviously counter examples, as with all these shallow matching techniques, this idea appears to work out reasonably well when it is used as one of the features in a machine learning approach.

2.3 Dependency relations

In part II of the project, we looked at a way to exploit syntactic information for RTE by computing the tree edit distance between the syntactic trees for T and H. It would of course be possible to simply use the tree edit distance as just one of the features in a machine learning approach to RTE. However, this has certain drawbacks. Not only is the computation rather expensive for large trees, it is also not easy to tune – or perhaps learn – the optimal insertion costs.

Fortunately there are other ways to include syntactic information. Recall from part II that a dependency analysis of a sentence consists of a set of

³http://cogcomp.cs.illinois.edu/page/software_view/4

$\langle head, relation, dependent \rangle$ triples. This implies that we can approximate the similarity between the dependency trees for T and H by the cardinality of the intersection of the sets of dependency triples for T and H respectively. Of course, we want a normalized version of this score. We also may want to assign a low weight to certain uninformative triples or even filter them out completely.

Remember how the BLEU score was essentially the average over shared word n-grams? A similar approach may apply to dependency triples where a single triple is analogous to a word bigram, two triples matching on either head or dependent (think: domino) are analogous to a word trigram, three matching triples to a 4-gram, and so on. This amounts to matching paths through the dependency trees of H and T respectively.

2.4 Polarity

Consider the following example:

- (4) T Drew Walker, NHS Tayside’s public health director, said: “It is important to stress that this is not a confirmed case of rabies.”
 H A case of rabies was confirmed.

This is a tricky case for string similarity measures, because the overlap between T and H is nearly perfect (completely at the lemma level), yet there is no entailment. The reason is of course the negation *not* in the Text. This contrast can be described as a difference in **polarity**: whereas the Text is a *negative* statement, the Hypothesis is a *positive* statement.

Suppose we define a set of negative words that toggle the polarity of a sentence: *Negatives* = {*not, refuse, wrong, deny, no, false, ignore, cannot, never, unsuccessfully*} This forms the basis for a very simple binary Negation feature, which simply counts the occurrences of negative words in both T (n_t) and H (n_h) and checks their parity:

$$NEG(T, H) = \begin{cases} 0 & \text{if } n_t \text{ and } n_h \text{ have the same parity} \\ 1 & \text{otherwise} \end{cases}$$

3 Further improvements

This Section lists a couple of other suggestions for possible improvements of an RTE system. This list is by no means exhaustive. There are many other paths to explore. An error analysis of your existing system may give you some ideas.

Text normalization Text normalization involves transforming all variation in expressions of time, date, currency, and so on, to a standard form.

This should evidently increase the chances of matching such expressions between H and T. For example, “200 dollar” does not match “\$200” or “two hundred dollars”.

String distance measures There are many other string distance measures to consider, such as the Levenshtein distance (or string edit distance), the Longest Common Subsequence or Cosine similarity.

Weighting IDF is just one form of weighting. Even for IDF, there are a number of alternative formulations (e.g. with or without taking the logarithm). Moreover our IDF values have so far been calculated on a relatively small document collection - the RTE development data - and using a much larger document collection is likely to result in better IDF weights.

Co-reference resolution In the example below, the entailment relation is clear to humans because they understand that *he* and *Omar Bakri* are one and the same person.

- (5) T: Britain said, Friday, that it has barred cleric, Omar Bakri, from returning to the country from Lebanon, where **he** was released by police after being detained for 24 hours
H: Bakri was briefly detained, but was released.

The computational task of figuring out that the anaphoric expression *he* refers to its antecedent *Omar Bakiri* is called **Co-reference Resolution**. Given this information, an RTE system has to infer that *he was detained* is equivalent to *Omar Bakiri was detained* (and analogously for *released*), which should facilitate matching to the Hypothesis. In terms of matching dependency triples, this would amount to deriving the triple $\langle OmarBakiri, obj, detain \rangle$ from the triple $\langle he, obj, detain \rangle$ and the identity $he = OmarBakiri$.

In practice, the accuracy of Co-reference Resolution systems is rather low and inferring consequences can be rather complex. However, since the Text is rather small, the number of possible antecedents is rather small, and hence co-reference is easier than in longer texts.

More training data “There’s no data like more data” is a famous quote attributed to Fred Jelinek. More training data from other RTE challenges is available on the web. Use all the development data you can get, but please stay away from test data. Notice however that there are differences between data sets due to changes to the task and the annotation procedure over the years. Also *preprocessed* versions may not be available.

RTE per task The pairs in the RTE data stem from four (related) NLP task: Information Extraction (IE), Information Retrieval (IR), Question

Answering (QA) and Automatic Summarization (SUM). Each `pair` element in the XML file has a `task` attribute with one of these four values. Some RTE systems have benefitted from taking the task information along in the form of an extra feature. Others went as far as building four different systems, each one tuned for a specific task.

Appendix: NLP resources

Here is a selection of some resources for NLP. This list is by no means extensive and searching the web will reveal many more resources. However, be wary of things that seem to be in an early stage of development or that are no longer supported, because these may be a dead end.

3.1 General NLP frameworks

These are some of the popular free open source NLP frameworks.

NLTK The Python Natural Language Toolkit is primarily intended for teaching natural language processing. It has many tutorials and extensive documentation. It contains many off-the-shelf tools and resources for processing English.

LingPipe LingPipe is another general NLP toolkit in Java. Excellent documentation and tutorials. <http://alias-i.com/lingpipe/>

GATE GATE is another NLP framework implemented in Java for developing industrial-strength applications. Rather complex and steep learning curve though. <http://gate.ac.uk/>

OpenNLP OpenNLP also hosts a variety of Java-based NLP tools based on maximum entropy machine learning. Software and documentation appears to be under development. <http://incubator.apache.org/opennlp/index.html>

3.2 WordNet

API's to Princeton's WordNet database are available for a wide range of programming languages - see <http://wordnet.princeton.edu/wordnet/related-projects/#local>.

NLTK has a Python API for retrieving lexical semantic relations from Wordnet. See http://nltk.googlecode.com/svn/trunk/doc/book/ch02.html#wordnet_index_term for a tutorial and <http://nltk.github.com/api/nltk.corpus.reader.html?highlight=wordnet#module-nltk.corpus.reader.wordnet> for an API documentation.

WordNet::QueryData is a Perl interface to the WordNet database. Available from <http://people.csail.mit.edu/jrennie/WordNet/>

3.3 Lexical semantic similarity

Ted Pederson's WordNet::Similarity is a frequently used Perl module that implements a variety of semantic similarity and relatedness measures based on information found in WordNet. See <http://www.d.umn.edu/~tpederse/similarity.html>. An online demo can be found at <http://talisker.d.umn.edu/cgi-bin/similarity/similarity.cgi>

NLTK offers similar functionality implemented in Python. See <http://nltk.github.com/api/nltk.corpus.reader.html?highlight=wordnet#module-nltk.corpus.reader.wordnet> for a API documentation.

3.4 Named Entity Recognition (NER)

LingPipe offers several NER implementations in Java and a complete model for NER in English. See <http://alias-i.com/lingpipe/demos/tutorial/ne/read-me.html> for instructions and <http://alias-i.com/lingpipe/web/models.html> for models.

NLTK has an NER implementation in Python. See http://nltk.googlecode.com/svn/trunk/doc/book/ch07.html#named_entity_recognition_index_term.

Stanford Named Entity Recognizer (NER) is another Java implementation with models for tagging in 3, 4 or 7 different classes. See <http://nlp.stanford.edu/software/CRF-NER.shtml>.

3.5 Text normalization

Java Number Normalizer is available from <http://u.cs.biu.ac.il/~nlp/downloads/normalizer.html>

3.6 Co-reference resolution

Stanford University offers a Co-reference Resolution system <http://nlp.stanford.edu/software/dcoref.shtml> as part of the StanfordCoreNLP Java package.

CherryPicker is another Co-reference resolution tool <http://www.hlt.utdallas.edu/~altaf/cherrypicker/>

See this post for some other options: <http://metaoptimize.com/qa/questions/1480/coreference-resolution-libraries>