IT3105 Project III: Recognizing Textual Entailment
Lecture notes for Part I

Erwin Marsi

Fall 2011

1 Introduction

Many things that are easy for computers are hard for us, for example, calculating the square root of 6745734692. Conversely, many things that we can do without a thought are extremely hard for computers. Many AI problems fall into this category. So does processing language. How is it possible that after decades of research by thousands of bright researchers with exponentially growing computational power, computers still can't do what any normal toddler can: learn a language?

Significant progress has been made in the computational treatment of language, in fields such as Natural Language Understanding, Computational Linguistic, and Natural Language Processing. Computers have become reasonably good at learning how to deal with structural aspects of language. For example, how complex words are composed from smaller parts (morphology) or how these words are combined into a grammatical sentence (syntax). Yet understanding the *meaning* of language is a problem that is far from being solved. Hence computers are still baffled by the language we humans use so effortlessly in our daily lives.

The meaning of language (*semantics*) and its use in particular contexts (*pragmatics*) have proven to be really hard to capture in strictly formal models of language. There is certainly a long tradition of logic models for the semantics and pragmatics of language in Philosophy, Linguistics and Computer Science (AI). Admittedly such models are elegant and insightful, but of limited use in practice. The two main problems from an applied point of view are insufficient *coverage*, lack of *robustness* and the *knowledge acquisition bottleneck.*

The problem with coverage is that most models cover only the well-organized, but rather rare aspects of language having to do with negation, quantification, modality, scope, etc. In practice, most real language use is rather disorganized and messy, full of phenomena that can not be accounted for by the models proposed. The changes of encountering the type of sentences favored by logicians – like, "*Every farmer who owns a donkey beats it.*" – are rather slim.

The second problem is that most formal models are too brittle. The input is assumed to be perfectly well-formed, standard language, adhering to the type of language rules that schoolteachers love. Unfortunately, small deviations from the standard – a spelling error, a missing or unknown word, an idiomatic expression, an unfinished sentence – cause the whole model to break down instantly. In contrast, robust models exhibit graceful degradation: continue to operate properly in the event of one or more the failures.

In addition, there is the problem that many theories about language are language-specific. This means that if you implement a knowledge-based model for handling, say, the grammar of English, that model will not work for Arabic or Chinese, because the grammars of these languages are very different. This means that for each different language, you will need to find – or even create – the knowledge required – that is, linguistic theory about the language – and derive and implement a computational model for it. Admittedly this has been done to some extent for a number of languages, mostly English, but it is infeasible for all languages in existence due to the huge efforts and costs involved. This problem is called the *knowledge acquisition bottleneck.*

At the end of the last century, partly in response to these problems, the field of Computational Linguistics and Natural Language Processing has shifted from rational, knowledge-based approaches to a new paradigm of empirical approaches to language. These are data-driven and rely on statistics and machine learning. Rather than implementing linguistic theory invented by linguists, computers learn language processing directly from large collections of examples, called *annotated corpora.* Learning usually takes the form of *supervised classification*, where a machine learning algorithm is trained on labeled examples to induce a model for this labeling task. However, unsupervised learning is gaining momentum as the web provides enormous amounts of unlabeled text for free.

The main reason this paradigm shift was simply that the empirical methods proved to be way more effective at solving real-world language processing tasks. A good example of this can be observed in *machine translation*, automatically translating one language into another. While in 70's and 80's machine translation was limited to a handful of language pairs due to the enormous costs of implementing a full fledged system for a given language pair, nowadays we have online services such as Google Translate providing translation for over 4000 language pairs. Even though translation quality is often far from perfect, the mere quantity and speed alone makes it an impressive accomplishment.

Against this background of the currently dominant empirical methods to language processing, this project is about applying empirical methods to semantics, that is, to understanding the meaning of language. This, however, not in the form of abstract models of certain aspects of semantics, but in the form of very concrete task: Recognizing Textual Entailment. In a nutshell, this means predicting if a given statement either does or does not follow from a given text. This task will be explained in more detail in the next Section. In the project, we will address this task in several ways. Initially we will use rather simplistic pattern matching techniques at the string level. In the second part, we continue with a more sophisticated structural matching technique. The third part applies generic supervised machine learning techniques to the task. The fourth and last part takes the form of a competition, in which you are challenged to implement a system that achieves the best possible performance – within reasonably limits, of course. Detailed requirements for these tasks can be found in the project description, available as `project-rte-description.pdf` from the project's website. The goal of these lecture notes, and the accompanying lectures, is to provide you with the task-specific knowledge you need to implement the four project parts.

2 Recognizing Textual Entailment

2.1 Textual entailment

Textual entailment is defined as a directional relation between two text fragments, termed T - the entailing *text*, and H - the entailed *hypothesis*. We say that T entails H if, typically, a human reading T would infer that H is most likely true. This somewhat informal definition is based on (and assumes) common human understanding of language as well as common background knowledge. Some examples of textual entailment, both true and false, are given in Table 1.

Entailment is a directional relation. The hypothesis must be entailed from the given text, but the text need not be entailed from the hypothesis. It is of course possible that the entailment relation is bidirectional, in which case T and H form a pair of *paraphrases*. For example, the sentences "*John is married to Mary*" and "*Mary is married to John*" entail each other and are therefore paraphrases of each other.

The definition of entailment allows presupposition of common knowledge, such as: a company has a CEO, a CEO is an employee of the company, an employee is a person, etc. For instance, in example 6, the entailment depends on knowing that the president of a country is also a citizen of that country.

The hypothesis must be fully entailed by the text. Judgment would be NO if the hypothesis includes parts that cannot be inferred neither from the text nor from the common knowledge. Thus even an evidently true Hypothesis such as "*Paris is the capitol of France.*" might not be entailed if the Text does not contain the information to support it (e.g., "*France announced that it will not support the latest EU proposals.*").

Table 1: Examples of textual entailment

Id	**Text**	**Hypothesis**	**Entails**
1	The drugs that slow down or halt Alzheimer's disease work best the earlier you administer them.	Alzheimer's disease is treated using drugs.	YES
2	Drew Walker, NHS Tayside's public health director, said: "It is important to stress that this is not a confirmed case of rabies."	A case of rabies was confirmed.	NO
3	Yoko Ono unveiled a bronze statue of her late husband, John Lennon, to complete the official renaming of England's Liverpool Airport as Liverpool John Lennon Airport.	Yoko Ono is John Lennon's widow.	YES
4	Arabic, for example, is used densely across North Africa and from the Eastern Mediterranean to the Philippines, as the key language of the Arab world and the primary vehicle of Islam.	Arabic is the primary language of the Philippines.	NO
5	About two weeks before the trial started, I was in Shapiro's office in Century City.	Shapiro works in Century City.	YES
6	Meanwhile, in his first interview to a Western print publication since his election as president of Iran earlier this year, Ahmadinejad attacked the "threat" to bring the issue of Iran's nuclear activity to the UN Security Council by the US, France, Britain and Germany.	Ahmadinejad is a citizen of Iran.	YES

Textual entailment differs from entailment in the strict logical sense. Cases in which inference is very probable (but not completely certain) are still judged as YES. In example 5, one could claim that although Shapiro's office is in Century City, he actually never comes in his office, and works elsewhere. However, this interpretation of T is very unlikely, and so the entailment holds with high probability. On the other hand, vague examples are avoided for which inference has some positive probability which is not clearly very high.

2.2 Recognizing Textual Entailment

Recognizing Textual Entailment (RTE) is the task of deciding, given T and H, whether T entails H. It is supposed to be carried out fully automatically by a computer program, without any human intervention.

The class of NO entailment can actually be divided into two subclasses. First, the Hypothesis can simply be *unsupported* by the Text, as in Example 4. Second, the Hypothesis can *contradict* the Text, as in Example 1. Some extension to the RTE task therefore formulate it as a three-way task, in which system must recognize entailment, contradiction or lacking support. Even though we are only corned with the traditional RTE taks in this project, it is worthwhile to remember this distinction, because it can often pay off to have a specialized part for recognizing contradictions in your system.

It should be emphasized once more that systems are supposed to recognize entailment, not to recognize true hypotheses regardless of the text.

2.3 Motivation

There are several reasons for why the RTE task is interesting and worthwhile to pursue. From a scientific point of view, the RTE taks provides an opportunity for empirical verification and comparison of a wide range of different approaches to natural language semantics. Earlier on, comparing different semantic theories and computational models was virtually impossible, because of differences in focus and goals, input and output, application domain, evaluation criteria and so on. The RTE task, however, is a (deceivingly) simple problem stated in terms of sentence pairs and makes hardly any theoretical assumption. Whether you want to tackle the task with low level pattern matching techniques or with a combination of logical analysis, knowledge bases and inference, any approach that gets the job done is fine.

Related to this point is that RTE data sets are randomly sampled from real text produced by human writers for completely different purposes, as in news papers, manuals or webpages. This means that there are no artificial doctored examples which were specifically created in support of a certain theory (recall the *donkey* sentences). Instead the data is to a large extent representative of everyday language use.

The task is also motivated from an applied point of view. Natural language allows us to express essentially the same meaning in many different forms. In other words, we can say the same thing in a lot of different ways. This happens to be one of the major obstacles in Natural Language Processing.

A case in point in Question Answering (QA). Suppose you have the following question:

(1) Who bought Overture?

Entering the question in a search engine, you get millions of hits and have to search the topmost results for the correct answer. It is the task of a QA system to do that job for you. It analyses the question and comes up with the format of an answer:

(2) X bought Overture

It then tries to find this pattern in the retrieved documents to determine the value of X. However, the documents may not contain literally contain this pattern, but instead may contain the same meaning in a different form. For example:

(3) Overtures acquisition by Yahoo

At this point a QA may hypothesize that this text entails the correct answer. The problem of finding a variant of the right answer can therefore be reduced to proving that the following entail relations holds:

(4)	T	Overtures acquisition by Yahoo
	H	Yahoo bought Overture

Similar arguments can be made for many other NLP applications, including Information Retrieval (IR), Information Extraction (IE), Automatic Summarization and Machine Translation. The conjecture posed by proponents of textual entailment is that many NLP tasks can be reduced to RTE. To what extent this is really true remains to be seen.

2.4 The RTE challenges

The Recognizing Textual Entailment (RTE) challenge is a competition (some people prefer the more friendly term *shared task*) that, starting from 2004, has been held yearly. Even though the exact task has varied over time, the common theme is that participants need to build a system that can recognize textual entailments. The challenges are organized in the following way.

At the start of the challenge, a development data set is released. This consists of few hundred Text-Hypothesis pairs that were annotated by human judges as cases of true or false entailment. Participants can use this dataset to develop their systems, to estimate accuracy, sometimes even to train or fine tune systems.

At the end of the challenge, participants receive a test data set. It has exactly the same format as the development set, except that the human entailment judgements are missing. The test is therefore *blind*, which prevents participants from cheating. For example, they can not tune their system on the test data. This is in line with best practice in machine learning experiments, where training and testing should be carried out on separate data sets to get a better estimate of how well the model performs on unseen data (generalization). Participants get limited amount of time, like 24 hours, to run their system on the test data and return the system output to the organizers in a specified format.

The organizers then compute accuracy scores by comparing each system's predictions to the true values. The true values are the entailment judgement by the same human annotators responsible for the entailment judgement in the development data. Participants receive the accuracy score of their system. Usually participants are allowed to submit multiple system with different settings.

Finally a workshop is organized about the challenge. The organizers present of overview and general analysis of the results. Participants with the best performing or most innovative systems present their work. Everything gets published in form of proceedings.

In the final part of this project, Part IV, is modelled after the RTE challenges. You build your system using the development data and use it to process blind test data at the end.

Figure 1 plots the accuracy scores on the test data of the participating systems over the course of five RTE challenges. Strictly speaking the scores over different years cannot be directly compared due to differences in the details of the task and the data sets. Across the board, however, average system performance ranges from 0.55 to 0.65 systems, whereas the best systems score between 0.7 and 0.8. This serves to show that there is no need to worry if your system does not have an accuracy close to 1.0.

2.5 Approaches to RTE

There are many different approaches to RTE. Since this is not lecture on RTE but on AI programming, we will only look at a couple of approaches. This does not mean that these are the best approaches. They just happen to pose interesting programming exercises.

In Part I of the project we look at lexical pattern matching techniques. This start at the most basic level of matching words, gradually improving the approach by including some linguistic information and a weighting scheme from Information Retrieval.

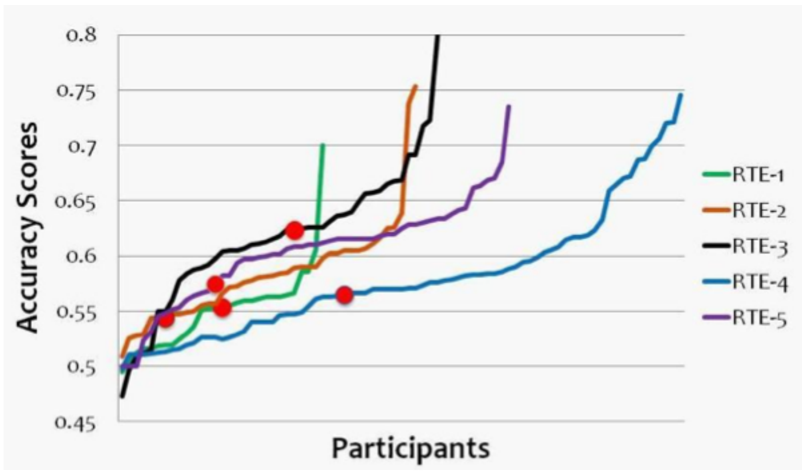

Figure 1: Accuracy scores per RTE challenge (from the RTE tutorial by Mark Sammons, Idan Szpektor V.G. Vinod Vydiswaran)

In part II, more linguistic information is exploited in the form of syntactic structures. We will implement a tree edit distance algorithm to measure the distance between Text and Hypothesis.

Part III is about applying generic supervised machine learning techniques to the task. Here we implement a system that extracts useful features and trains a machine learner to predict entailment relations.

Part IV is a mini-RTE challenge in which you are challenged to implement a system that achieves the best score on a blind test set.

3 Lexical pattern matching

3.1 Word matching

Most approaches to RTE rely on matching the Hypothesis to the Text. The idea is that if the Hypothesis can be matched to, aligned to or otherwise shown to closely resemble (parts of) the Text, then an entailment relation is likely. The most basic way of accomplishing this is by simply measuring how many words from the Hypothesis also occur in the Text. In other words, measuring the overlap in words.

Counting the number of shared words alone is not a good indicator of entailment, as a longer Hypothesis is more likely to have shared words with the Text. Hence this count needs to be normalized for differences in sentence length, which can be accomplished by dividing by the total number of words in the Hypothesis. This leads to the following formulation of *Word Match*.[1]

[1]Incidentally, as we will see in Section **??**, this corresponds to the unigram precision.

$$Word\ Match = \frac{\#words\ in\ both\ H\ and\ T}{\#words\ in\ H} \tag{1}$$

The only thing left to do is to set a threshold value on the Word Match value above which entailment is true and below which entailment is false. The Ietoptimal threshold value can be determined by iterating over a range of possible values and calculating the accuracy score on the development data.

3.2 Matching lemma and part-of-speech tag

Clearly word matching is a very crude method that fails miserably in recognizing certain cases of entailment. Some improved can be gained by a little bit of natural language processing.

Lemma Consider the following example in which an entailment relation evidently holds.

(5) T: He is looking out of the windows
H: He looks out of a window

Conceptually there is a perfect match, but unfortunately the finite verb *looks* does not match the gerund *looking* and neither does the singular noun *window* match its plural form *windows*. How can we make this work?

If you look up a word in a dictionary, changes are that you will not find it in the exact form your are looking for. Instead many dictionaries only contain the *lemma*, or base form of a word. For example, the surface forms *to look*, *looks*, *looked*, looking all share the same lemma *look*. Likewise, *window*, *windows* share the same lemma *window*.

Mapping the words from Example (5) to their lemma yields:

(6) T: He be look out of the window
H: He look out of a window

This transformation reveals that the Hypothesis matches the Text perfectly in terms of lemmas. It is easy to define a Lemma Match measure analogously to Word Match measure.

Lemmatization – the process of mapping words tobase their lemmas – is fairly easy for English, because the number of surface word forms is rather limited and a simple lookup procedure solves most cases. However, for other languages with a more complex morphology (word structure), lemmatization can be a really hard problem. Fortunately, we have a preprocessed version of the RTE data which provides us with lemmas, so the only thing we need to do is extract the lemmas for Text and Hypothesis, compute the Lemma Match score, and determine an optimal threshold.

Part-of-speech Consider the following example:

(7) T: The lead singer left to wave at us
H: The left wave threw lead at us

Even though the Hypothesis shares most of its words with the Text (6 out of 7), there is definitely no entailment relation. At least three of word matched are wrong:

1. *lead* in the sense of "first" does not match *lead* in the sense of "metal"

2. *left* in the sense of "going aways" does not match *left* in the sense "opposite of right"

3. *wave* in the sense of "greeting" does not match *wave* in the sense of "movement in fluid"

Such words with identical spelling but different meanings are called *homographs*. Homographs can often be distinguished by word class, especially in English. *Part of speech* (POS) is the linguistic term for the categories of a word. Without going into too much detail, some of the major words categories in English are Noun (N, e.g., *car*), Verb (V, e.g., *drive*) Adjective (A, e.g., *blue*) and Adverb (Adv, e.g., *quickly*).[2] The process of classifying words according to a predefined set of word categories is called *part-of-speech tagging*, which is a basic processing step in many Natural Language Processing tasks. Tagging the words from example (7) would give something like:

(8) T: The/Det lead/N singer/N left/V to/Aux wave/V at/Prep us/Pro
H: The/Det left/A wave/N threw/V lead/N at/Prep us/Pro

The important thing to notice is that the POS tag allows us to distinguish the different senses of the homographs *wave* – Verb in the Text versus Noun in the Hypothesis) – and *left* – Verb in the Text and Adjective in the Hypothesis.

The preprocessed version of the RTE data contains POS tags, so again the only thing we need to do is extract the word plus POS tag combination for Text and Hypothesis, compute a Word + POS Match score, and determine an optimal threshold.

Admittedly, homographs are infrequent in real text, which includes the RTE data, so do not expect significant improvements in accuracy from this.

[2]There is actually little agreement among linguists about the number of word categories for a specific language, let alone for all languages taken together. Some coarse-grained POS tag sets consist of no more than a couple of tags, whereas fine-grained ones may run into hundreds of tags.

3.3 Inverse document frequency weighting

Some words are more equal than others. Search engines are very aware of this. If you type a query like "*the husband of the queen of the Netherlands*", all instances of *the* and *of* are most likely completely ignored, because they are too frequent and lack sufficient meaning to help the search. One way to accomplish this is by means of list of *stopwords*, which filtered out of the query. Yet even among the remaining words, some may be more important to the search. Search engines therefore usually assign a weight to the different terms of query. One very common weighting scheme is TF*IDF (Term Frequency times Inverse Document Frequency). Here we will only consider the IDF part.

Inverse Document Frequency (IDF) IDF is defined over a collection of documents. The IDF weight of a word is simply the inverse of the number of documents that the word appears in. If a word occurs in many documents from the collection, i.e. its document frequency is high, than its inverse document frequency is relatively low. In contrast, if a word occurs in only a few documents, its inverse document frequency is relatively high. Therefore words with a high IDF value are good indicators for specific documents, whereas words with a low IDF value generally do no help to discriminate among documents. Hence weighting words by their IDF value has often been found to improve search results.

IDF for RTE The same type of IDF weighting can also be applied to word matching for RTE. In this setting, we consider each Text and Hypothesis to be a document, together constituting the document collection.. IDF values are computed for all words occurring in this document collection. Next we define the following IDF Word Match measure[3]

$$IDF\ Word\ Match = \frac{\sum_{w \in T \cap H} IDF(w)}{\sum_{w \in H} IDF(w)} \tag{2}$$

and go about the same way to find an optimal threshold value.

3.4 BLEU algorithm

The BLEU (BiLingual Evaluation Understudy) algorithm was originally developed for machine translation. [**?**].[4] Earlier evaluation of machine translation systems was always dependent on human judgements of the translation

[3]Note that the set theoretic statement, as in $w \in T \cap H$, is an abuse of notation, because if T and H would really be sets, we can not account for the fact that the same word can occur multiple times in a T or H. The proper way to formalize this would be in terms of *multisets* or *bags*.

[4]The description of BLEU below is based directly on this source, copying some parts of the text literally.

quality. This is expensive and time-consuming and therefore constituted a serious bottleneck for progress. BLEU was proposed as a method for scoring the quality of machine translation output in an automatic way, without any human intervention. It therefore allows quick evaluation and optimization of machine translation systems.

BLEU essentially measures the similarity between a system translation and one or more human reference translations. The more the two are alike, the higher the BLEU score. Several researchers have therefore proposed to use BLEU, or a modified version of BLEU, to score the similarity between hypothesis and text.

N-grams An *n-gram* is a subsequence of n items from a given sequence (for $n >= 1$). In the case of sentences consisting of words, an n-gram is thus a subsequence of n words from a sentence. For example, given the sentence

```
There is a cat on the mat
```

the list of all its 2-grams (*bigrams*) is

```
There is
is a
a cat
cat on
on the
the mat
```

and the list of all its 3-gram (*trigrams*) is

```
There is a
is a cat
a cat  on
cat on
on the mat
```

The list of all 1-grams (*unigrams*) is simply the list of all words. Some n-grams may occur more than once in a text, in which case it makes sense to talk about *n-gram counts.*

Shared n-grams There are usually many good translations of a given source sentence. Yet humans can distinguish good from bad translations. Consider the following two candidate translations of a Chinese source sentence.

(9) C1: It is a guide to action which ensures that the military always obeys the commands of the party.

C2: It is to insure the troops forever hearing the activity guidebook that party direct.

Although about the same subject, they are very different in quality. Three reference human translations of the same sentence appear below.

(10) R1: It is a guide to action that ensures that the military will forever heed Party commands.
R2: It is the guiding principle which guarantees the military forces always being under the command of the Party.
R3: It is the practical guide for the army always to heed the directions of the party.

Clearly the good translation C1 shares many words and phrases with these three reference translations, while the bad translation C2 does not. Observe that C1 shares "It is a guide to action" with R1, "which" with R2, "ensures that the military" with R1, "always" with R2 and R3, "commands" with R1, and finally "of the party" with Reference 2 (all ignoring capitalization). In contrast, C2 has far fewer matches and their extent is less.

From this example, it is clear that a program can rank C1 above C2 simply by comparing n-gram matches between each candidate translation and the reference translations. The primary task of BLEU is to compare n-grams of the candidate with the n-grams of the reference translation and count the number of matches. These matches are position-independent. The more the matches, the better the candidate translation.

N-gram precision The basis of the BLEU metric is the well-known precision measure. For simplicity, we initially consider unigrams (words). To compute precision, sum the number of candidate translation unigrams which also occur in any reference translation and then divide by the total number of words in the candidate translation. Thus C1 achieves a unigram precision of 17/18, whereas C2 achieves a unigram precision of 8/14.[5]

N-gram precision is computed similarly for any n: sum the number of candidate translation n-grams which also occur in any reference translation and then divide by the total number of n-grams in the candidate translation. Thus C1 achieves a bigram precision of 10/17, whereas the lower quality C2 achieves a bigram precision of 1/13.

This sort of n-gram precision scoring captures two aspects of translation: adequacy and fluency. A translation using the same words (1-grams) as in the references tends to satisfy adequacy in the sense that it adequately

[5]The original BLEU description included a *modified* n-gram precision measure, which prevented certain ways to fool the BLEU measure. However, since this is irrelevant in an RTE context, we will abstract from it here.

conveys the information in the original source sentence. The longer n-gram matches account for fluency in the sense that the translation is likely to be a grammatically well-formed sequence of words.

Original BLEU algorithm In order to capture both adequacy and fluency, we take an average over the n-gram precision scores. However, since the likelihood of finding a matching unigram (word) is far greater than the likelihood of finding a matching 4-gram, the modified unigram precision is much larger than the modified bigram precision which in turn is much bigger than the modified trigram precision, and so on. As a consequence, the arithmetic mean tends to be dominated by unigram precision, and therefore fails to represent fluency. Therefore BLEU uses the average *logarithm* with uniform weights, which is equivalent to using the *geometric mean* of the n-gram precisions.[6] This leads to following definition

$$mean\ precision = \sqrt[n]{p_1 \times p_2 \times \ldots \times p_n} \tag{3}$$

which is equivalent to

$$mean\ precision = exp\left[\frac{1}{N}\sum_{n=1}^{N} log\ p_n\right] \tag{4}$$

Finally, the BLEU score is calculated according to Equation 5.

$$BLEU = BP \times mean\ precision \tag{5}$$

This includes a so-called *brevity penalty* factor BP. Its purpose is to penalize translations that are shorter than the reference translations, in a way to compensate for a lack of recall. However, since the hypothesis sentence is always shorter than the text sentence(s), applying a brevity penalty does not make a lot of sense for recognizing entailment. We will therefore not discuss the brevity penalty any further here.

In practice, the contribution of n-gram precision for $n > 4$ turns out be negligible, so by default the BLEU score is computed for n in the range from 1 to 4.

Modified BLEU algorithm As already mentioned, applying a penalty for brevity does not work in the context of RTE. It turns out that there are other problems with the original BLEU algorithm when applied to RTE. One of these stems from the fact that BLEU is intended for evaluating a single system translation against *multiple* human reference translations. However, the RTE context provides only one "reference", that is, the text. As a result, n-grams from the hypothesis are less likely to match those in the text and in

[6]See the original paper for more elaborate argument along this line.

fact n-gram precision is often zero. And if any of the n-gram precision scores is zero, the mean precision becomes zero and the BLEU score becomes zero!

One way to resolve this issue is to take the arithmetic rather than the geometric mean. Thus we end up with Equation 6.

$$Modified\ BLEU = \frac{1}{N}\sum_{n=1}^{N} p_n \tag{6}$$

Even though this reduces the influence of higher n-grams, and therefore the importance of fluency, this approach appears to work better in practice.

Using BLEU to predict entailment In order to use BLEU for RTE, simply apply the modified BLEU algorithm to calculate a BLEU score, taking the hypothesis as the candidate translation and the text as the reference translation. The only preprocessing required is to lower-case all words. Set a threshold value on the score above which entailment is true and below which entailment is false. You can determine the optimal threshold value by iterating over a range of values and calculating the accuracy score on the development data.