Textual Entailment

Christoph Stenkamp

University of Osnabrueck, 49074 Osnabrueck, Germany
cstenkamp@uos.de

Abstract. *Textual Entailment* techniques try to answer the question if some natural-language hypothesis holds, given a natural-language premise. While generally being considered an in-between step for other NLP problems such as Machine Translation or Question Answering, some argue that correctly identifying textual entailment is a sufficient condition for true Natural Language Understanding. This paper clarifies the topic, explains some applications and summarizes a few past and present techniques to approach the problem. Whereas the past approaches are very divergent in their methodologies, they led to only limited success. With the creation of vast datasets, almost all modern techniques rely on artificial neural networks and high-dimensional word embeddings. Throughout describing the different approaches, a focus will lie on the approach of using *natural logic* as well as the two main neural network architectures.

1 Introduction

The task of *Recognizing Textual Entailment* (RTE), also known as *Natural Language Inference* (NLI), is the problem of determining whether a natural language *hypothesis* can be reasonably inferred from a natural language *premise*.

While the huge field of logic-based AI deals with the related topic of *formal deduction*, NLI constitutes a different problem altogether – instead of demanding chains of formal reasoning or even strict logical consequences, an NLI inference is considered valid if an average person would likely accept the hypothesis, given that the premise holds.

While the usage of thesauri like WordNet[1] makes it trivial to detect entailment on the word level, doing the same on the sentence level is a far more complex task, as there is an infinite amount of sentence-pairs to consider.

Starting with the *PASCAL Recognizing Textual Entailment (RTE) Challenge* in 2005 [10], scientific interest in this field has surged, growing from a niche to a very active research field with a number of issues, conferences and challenges dedicated to it.

Solutions to NLI range from shallow approaches based on word-level similarities, over more advanced methods that consider syntax or lexical databases, to sophisticated solutions relying on formal logic and problem solvers. All of these techniques have different advantages and shortcomings, which this paper will

[1] Fellbaum, C. (1998). WordNet: An Electronic Lexical Database. MIT Press.

elaborate on. With the creation of the *Stanford Natural Language Inference corpus* (SNLI, [8]) in 2015 and the Multi-NLI corpus [7] in 2017, approaching the problem with artificial neural networks has become new state-of-the-art. This paper will also take a look at such techniques, further providing an outlook of how the field may evolve in the future.

Reading Instructions The remainder of this paper is structured as follows: In the first part of **section 1**'s remainder, some clarifications are made to describe the problem and its ramifications in appropriate detail, before giving an overview of why this topic is relevant to the scientific community. Afterwards, some re-occurring problems are listed and commented on. **Section 2** provides an overview of historical approaches, starting with simple text-based systems, before describing complex logic-based systems. Furthermore, a detailed description of *natural logic* is provided. **Section 3** provides an overview over modern, neural approaches. After clarifying the background for these, it will describe sentence-encoding based models, before moving over to attention-based systems. The section closes with a short discussion of neural approaches. **Section 4** ends this paper with the conclusion.

1.1 Clarifications

The Definition of Entailment While entailment has a clear definition in set theory and formal logic, the same cannot be said about natural language. Classical linguistics differentiate between *entailment*, *conversational implicature* and *conventional implicature/presupposition* (cf. [17]), Not only does the definition of textual entailment certainly include presuppositions and implicatures, but it further assumes common human understanding of language and background knowledge.

World Knowledge and Word Knowledge Optimally, recognizing textual entailment would only require linguistic knowledge, without any world knowledge. There is, however, a continuous linguistic debate on how those two are separable, if they are at all (cf. [17], [9]). To quote an example of such from [9]: *"[I]f I do not know that belladonna and deadly nightshade name the same plant, I will not recognize that an instruction to uproot belladonna entails an instruction to uproot deadly nightshade"*. While this is arguably a failure of botanical knowledge and not language understanding, many sample pairs for textual entailment rely on such knowledge. According to Dagan et al. [10] who created the RTE-dataset for the first PASCAL RTE challenge, 37% of the pairs in this corpus can be solved purely syntactically, and 49% can be solved utilizing a thesaurus like WordNet.

Recognizing, Finding and Generating. According to Androutsopoulos and Malakasiotis (2009), Textual Entailment poses three challenges that can be addressed. A textual entailment *recognizer* receives a pair of two sentences as input, returning a judgment whether the pair constitutes a valid textual entailment pair. An entailment *generator* receives only one sentence as input, outputting a set of expressions that are entailed by this sentence. Finally, a textual entailment *extractor* receives as input an entire corpus of texts, having the job to return pairs of language expressions that constitute correct entailment pairs. While the latter is arguably very useful for automatic text summarization, this paper will, due to space limitations, focus only on Textual Entailment *Recognizers*.

1.2 Significance and Necessity of the Topic

Natural Language Inference is generally seen as an in-between task to test a system's language processing abilities. It is generally understood, that correctly identifying inference is a necessary condition for a system to have general *Natural Language Understanding* (NLU) capabilities[2]. MacCartney and Manning [2] even argue that one might see it as a sufficient condition.

The standard examples of what NLI can be used for include Machine Translation, Question Answering (QA), Dialog Systems, Automatic Summarization, Paraphrase Detection and Relation Detection (cf. [10], [9], [6], [4]). According to Williams et al. [7], those systems depend in large part on the success of their ability to recognize natural language inference. The corpora of the third and fourth RTE-challanges [10] consited for example of problems borrowed from the domains of question answering, information retrieval, information extraction, and multi-document summarization.

Question Answering Many original methods for textual entailment were developed having QA-systems in mind. In general, the task of QA is to return a textual expression, extracted from a large collection of documents, providing a good answer to a question posed in natural language.

While the relation between a question and its answer can be cast in terms of logical entailment (see [12]), RTE can enable a QA system to identify correct answers with increased accuracy. Harabagiu and Hickl [12] provide a number of ways how an NLI recognizer can be incorporated into a QA system, stating that the accuracy of their QA system increased by up to 20% when using textual entailment to filter or rank answers.

Another straight-forward approach to incorporate NLI in QA-systems is to check if a candidate answer is textually entailed by any portion of the part of a text corpus retrieved by the system (cf. [14]). Condoravdi et al. [9] provide further detail about how incorporating NLI into their Eureka-system proved useful for answering yes/no questions.

Even though incorporating RTE can clearly increase overall performance in a number of tasks, there are currently very few articles reporting actual applications of NLI methods. This is despite the fact that there is a mass of articles proposing new methods, testing them in vitro and very often pointing to possible applications (as also noticed by [1]).

[2] According to [8], *nearly all questions about meaningfulness in language can be reduced to questions of entailment and contradiction in context.*

1.3 Problems in NLI

Some of the problems an NLI system must deal with include:

- **Polarity differences**, as *"this is not a confirmed case of rabies"* vs. *"a case of rabies was confirmed"* (cf. [1])
- **Modality differences** *"a case may have been confirmed"* vs. *"a case has been confirmed"* (cf. [1])
- **Synonymym, Hyponomy**, or **Antonomy** of words (cf. [7])
- **Named entity recognition**, as well as **relation of time and space expression** (cf. [11])
- Correctly identifying **coreferentials** (cf. [11])
- **Temporal Reasoning** and **Causal reasoning**, as *Khan sold nuclear plans* $\Rightarrow$ *Khan possessed nuclear plans* (cf. [2])
- **Paraphrases**, as *McEwan flew to Rome* $\Rightarrow$ *McEwan took a flight to Rome* (cf. [2])
- **Relation extraction** (*Bill Gates and his wife, Melinda...* $\Rightarrow$ *Melinda Gates is married to Bill Gates*) (cf. [2])
- **Quantification**, **Belief**, and **lexical and syntactic ambiguity** (cf. [7])

Logic vs Natural Language. The challenges of NLI are quite different from those encountered in formal deduction: the emphasis is on informal reasoning, lexical semantic knowledge, and variability of linguistic expression.

If one represented NL expressions by a logical formula (as eg. *first-order logic* (FOL)), one may want to think of textual entailment in terms of logical entailment: p entails h iff $(\mathrm{FOL}(p) \wedge B) \models \mathrm{FOL}(h)$, where B is a knowledge base.

A sample textual entailment problem could thus be converted into the following FOL representation (taken from [1]):

Premise	Leonardo da Vinci painted the Mona Lisa.
Premise in FOL	$isPainterOf(DaVinci, MonaLisa)$
Hypothesis	Mona Lisa is the work of Leonardo da Vinci.
Hypothesis in FOL	$isWorkOf(MonaLisa, DaVinci)$
Knowledge Base B	$\forall x \forall y : isPainterOf(x, y) \Rightarrow isWorkOf(y, x)$

Applying a sound and complete reasoner for first-order logic could be used to confirm that this entailment holds, and thus that the premise indeed textually entails the hypothesis.

A number of problems obviously arise from that definition: For example, many natural language phrases are ambiguous and cannot be translated into a single logical formula. For those that could in theory be translated, the problem arises that there is not (and probably cannot be) a system that properly does translate any given statement into such a language. Even with such a system, there is no logic system that corresponds exactly to human language and if it would exist, it is not given that it would be decidable or that a proof can be constructed in the given amount of time.

Ambiguity and Referencing Problems Many natural language sentences are ambiguous or contain referencing problems. This of course impacts RTE, as a semantic relation may hold under some interpretations, but not for others. A further problem arises in reference resolution (in the broad sense, see [9]), or detecting if two texts talk about the same entity – Entailment and contradiction relations often presuppose a shared subject, and a working system must correctly identify if that is given.

2 Overview of Historical Approaches

According to [6], traditional approaches to NLI are either feature-rich models, that leverage machine learning techniques on hand-crafted features of both premise and hypothesis, or formal reasoning methods, that convert a sentence into a logical representation, to subsequently search for a model or a proof. Hickl et al. [11] subdivide approaches working with logical representation further into approaches that use an inference mechanism on the respective representation language, and ones using the classical AI definition of entailment, building models of both premise and hypothesis to check if the models of the hypothesis are in fact a subset of those of the premise.

In the following sections, I will elaborate on different approaches for Natural Language Inference that do not rely on vast datasets, relying either on hand-crafted features or on a logical formalism. These approaches use the RTE-corpora to measure their performance, achieving accuracies between 50.4% (slightly above baseline) and 75%.

2.1 Text-Based Systems

The easiest way to detect textual entailment is to work directly on the surface strings of both premise and hypothesis. For that, these systems often follow the classical linguistic preprocessing pipeline, such as part-of-speech tagging or named entity recognition (cf. [1]). Afterwards, different measures of *string-distance* are used to check if an inference holds. If these measures (or a combination of those) fall under a certain threshold, the pair is considered a valid inference.

One example of such a distance is the *string edit distance* (also known as *Levenshtein distance*), that counts the minimal number of operations (insertions, deletions or substitutions) to transform the premise into the hypothesis. To account for the asymmetry of the entailment relation, different weightings can be given to respective operations (cf. [2]). A classifier using only this edit distance achieved an accuracy of 54.9% on the RTE-4 corpus (cf. [1]).

Another distance measure for text is the *Bilingual Evaluation Understudy (BLEU) Score* (Papineni et al. 2002). This measure counts matching n-grams of the hypothesis (independent of their position) to n-grams in the reference text, counting each occurrence only once. While being unaware of synonyms and paraphrases, it does seem to provide reasonable information – a classifier using only the BLEU score, length difference and word overlap count achieved an accuracy of 50.4% (baseline being 33.3%) on the SNLI corpus.

As the premise is often much longer than the hypothesis, many text-based systems employ a sliding window over the latter. If a part of p's surface string is similar (via a distance measure) to the hypothesis, it is an indication that it may be entailed. Thus, it makes sense to use the largest similarity score of such sliding windows and the premise. In many correct textual entailment pairs, however, using a single sliding window of a fixed length may still be inadequate, because h may correspond to several non-continuous parts of p.

A system that employs multiple distance measures is for example the one used by [14] for the third RTE challenge. Their system uses string similarity at the lexical and shallow syntactic level. For that, it employs ten different string similarity measures (next to the mentioned Levenshtein distance for example also the 3-gram distance, number of words both strings have in common and more). Those measures are not only run on the original word pairs, but also on processed versions of the respective strings, as for example their stems, part-of-speech-tags or strings consisting of only the nouns of the original sentences. To correctly handle strings of different length, they also computed the mentioned measures for all substrings of the premise of the same length as the hypothesis. They used the development-set of the RTE-3 corpus to train a classifier using a Support-Vector-Machine (SVM), selecting only the features that helped the most.

Their system reached an accuracy of 61.75% on the test set of the RTE-3 corpus, but 73.5% on the subset of those pairs corresponding to likely word pairs for a Question Answer-systems, which is what they tuned their model for.

2.2 Logic-Based Systems

As mentioned in section 1.3, one can also try to solve the textual entailment problem by translating premise and hypothesis into corresponding logical formulas. In practice, these approaches have however many shortcomings, as it is not only difficult (and in parts impossible) to translate the sentence, but also to formulate a reasonably complete knowledge base.

A partial solution may be to obtain common sense knowledge from resources like WordNet: Since "assasinate" is a hyponym of "kill" in WordNet, an axiom like $\forall x \forall y : assasinate(x, y) \Rightarrow kill(x, y)$ could be generated and added to the knowledge base (cf. [1]).

A system that uses both of the previously mentioned logic-based techniques (i.e. trying to solve and building models) is that of [13]. Their method is a hybrid, combining a shallow text-based system (using a classifier based on word overlap after tokenization and lemmatization of the input strings) with a logical system. The required knowledge base B for the logical system is derived on-the-fly using WordNet hypernyms as well as generic axioms and geographical knowledge.

Their logical representation is built as follows: A CFG-Parser is used to get the semantic representation of each pair, which is then further processed into a first-order fragment of the DRS-languge used in the Discourse Representation Theory (cf. [13]), and subsequently into FOL. To check whether entailment holds, they used a first-order-logic theorem prover that tries to prove the hypothesis

$(p \wedge B \rightarrow h)$, as well as a model builder that tries to find a model for the negation of the input. As these methods require an unrealistic amount of background knowledge, the system also incorporates a way to find approximate entailments. For that, they compute the model size of $B \wedge p$ and $B \wedge p \wedge h$, with the idea that a small number likely corresponds to an entailment.

To combine their deep and shallow approaches, all the generated information is combined to yield a decision tree classifier. While the performance of this classifier is with 56.25% accuracy far from good, their error analysis shows some interesting results: The system was able to create semantic representations for 774 of the 800 sentence pairs in the dataset, achieving a coverage of 96.8%. However, only 30 proofs where found by the system. Of these 30 proofs, seven where incorrect and occurred due to a lack of background knowledge and lexical semantics.

2.3 Natural Logic

The aforementioned approaches either relied on very simple techniques like measures of word overlap, or sophisticated first-order-logic theorem provers, that have a relatively high precision for the sentences they can translate into FOL, but a very poor recall as only a tiny fragment natural language sentence could properly be translated and combined with an appropriate knowledge base.

The idea of *natural logic* (MacCartney 2009, [3]) is supposed to be a middle ground – it is a theorem prover for sentences written in natural language, without the loss of recall by translating into FOL. Natural logic tries to prove that the hypothesis follows from the premise by incremental, conditioned edits to the expressions in natural language.

Natural logic aims to explain inferences involving monotonicity[3], in which semantic elements of a sentence can be truth-conservingly expanded or contracted. The sentence *"Every meal without wine is a terrible crime"* can thus truth-conservingly be relaxed to *"A dinner without drink is a crime"*, which is why the latter sentence is entailed by the former.

In natural logic, entailment is a *containment* relation, where the entailment property of two sentences breaks down to the entailment of their parts. For that, it follows the classical Fregean compositionality principle: a sentence is a truth value t, consisting of entities e and functions $\langle \alpha, \beta \rangle$ (where α and β in turn are possibly complex constructs of entities, functions and truth values). A sentence is entailed by another if all parts are entailed by their respective counterparts – for truth-conditions, $c \sqsubseteq d$ iff $c \rightarrow d$, for entities $c \sqsubseteq d$ iff c is a subset of d, and for functions it must hold that if the one function is fulfilled, the other one is necessarily fulfilled as well. With that, entailment relations between many semantic types can be introduced. For example, $today \sqsubseteq this\,month$, $French \sqsubseteq European$, or $everybody \sqsubseteq somebody$. In most cases, dropping a modifier is also truth-preserving: $eat\,quickly \sqsubseteq eat$.

[3] Monotonicity refers to the property of some logical calculi or natural language, that one can add additional premises to any statement such that the conclusion still follows.

Further, it is important to detect *upward-monotonicity* in contrast to *downward-monotonicity*: While generally a sentence p entails a sentence h iff all of p's parts entail h's parts (upward-monotonicity), some linguistic constructions are downward-monotone (like negations, restrictive verbs (like $fail$) or certain adverbs (eg. *without*)), and some expressions (like *some* or $former$) indicate the complete lack of monotonicity. When processing a sentence, one must first generate a constituent-tree from it, start with the monotonicity directions of the leaves and then project and compose them upwards via functional application, to get the effective polarity of a single semantic element in relation to its sentence.

The created system, *NatLog*, thus works as follows: It consists of three steps, namely linguistic preprocessing, alignment and finally entailment classification. The preprocessing is a minimal version of the classical-linguistics pipeline: The words of *premise* and *hypothesis* are tokenized, part-of-speech tagged, and phrase-structure-parsed using the Stanford parser[4]. Afterwards, the constituents are marked with respect to their monotonicity as explained above.

In the second step, an alignment is established between p and h. For that, *NatLog* finds a minimal sequence of atomic edits (insertion, deletion and substitution) of constituents to construct the hypothesis from the premise, keeping track of *light edits* like only changing an article.

In the final step, the alignment problem for the sentence is solved by decomposing it into an alignment for every edit. For that, a trained classifier (namely a decision tree) is used to predict one of five entailment relations between the edit, namely *equivalence* ($p = h$), $forward$ ($p \sqsubset h$), *reverse* ($p \sqsupset h$), *independent* ($p \not\sqsupseteq h$, $p \not\sqsubseteq h$) and *exclusive* ($p \sqsupseteq \neg h$). While the former keep entailment, *reverse* and *independent* are unknown (in the RTE challenges interpreted as non-entailment), and the latter a clear non-entailment. On the word-level, next to the previously explained entailment relations, WordNet is additionally used to get lexical information on synonymy, hyponymy and antonymy. The resulting entailment classifiers are then combined to yield the classification for the whole sentence. An example classification is given in Figure 1.

While NatLog works appropriately in its domain, there are many aspects not covered by it, as temporal or causal reasoning, relation extraction or the recognition of paraphrases (see sec. 1.3). Further, NatLog requires a straightforward edit distance which is not given in many RTE problems, which is why NatLog cannot be used as a general solver for them.

For these reasons, MacCartney and Manning [2] care mainly for the system's *precision* for its *yes*-predictions when tested on the PASCAL RTE3 challenge dataset. While the system's accuracy with 57.38% is not really good, their precision of 68.06% on the test set and 76.39% on the development set appears more compelling. In general however, the monotonicity-related inferences have only limited applicability in general natural language inference.

[4] See *Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In Proceedings of ACL-03*

Sentence pair:

Premise	*An Irishman won a Nobel prize.*
Hypothesis	*An Irishman won the Nobel prize for literature.*

Edits:

type	edit	monotonicity	entailment relation
substitute	$a \to the$	$\uparrow$	$=$
insert	*for literature*	$\uparrow$	$\sqsupset$

top-level inference:
$= \circ \sqsupset \Rightarrow \sqsupset \Rightarrow$ no entailment

Fig. 1. The operation of NatLog's entailment model on an example sentence.

2.4 Other Systems

There are many other systems, based on different approaches, that could be mentioned in a paper like this. Some of them rely on symbolic meaning representations (possibly incorporating databases like WordNet), text transformation rules and much more.

One system worth mentioning is for example the *LLC Groundhog* system [11], incorporating many of the aforementioned techniques into a single classifier, trained via a specifically collected corpus over 200k newswire articles. Doing so, Groundhog achieved an outstanding accuracy of 75% in the second PASCAL RTE challenge.

Unfortunately, spatial limitations do not allow to go into more detail of these systems, which is why it is referred to the respective paper, or to [1], who provide a good overview of systems until the year of that paper's writing (2009).

3 Overview of Neural Approaches

In 2015, Bowman et al. created a new dataset for NLI, termed the *Stanford Natural Language Inference* (SNLI) corpus. With over half a million samples, this dataset allowed for neural architectures to enter the stage of recognizing textual entailment.

3.1 Background for Neural Approaches

Datasets

SNLI Due to the mediocre performance of hand-crafted NLI systems, and following the success of ANNs in many NLP problems, Bowman et al. (2015) had the objective of empirically analyzing learning-centered approaches to NLI. While there were already previous corpora for NLI, these were either to small for training data-intensive models (as the PASCAL RTE1-4 corpora [10], which contained only 800 items in development- and test-set), or of poor quality (as the corpus of [11], who used newswire headlines and their corresponding first sentence, obviously introducing a certain inductive bias).

To collect their data, [8] employed a crowdsourcing framework using Amazon Mechanical Turk, in which humans were instructed to provide three sentences to a given premise, one being *necessarily true* given the premise, one being *necessarily false*, and one being *possibly true*. Their dataset thus consists not only of the categories *entailment* and *contradiction*, but additionally contains the category *neutral*[5]. Asking the workers to produce their hypotheses completely free allowed for richer examples than in previous datasets, making string-editing techniques as for example *NatLog* (chapter 2.3) harder and less accurate. The whole SNLI dataset consists of 570k pairs.

To circumvent referencing problems, the tasks were grounded in specific scenarios, where both premise and hypothesis are supposed to be descriptions of an image. In fact, the premises of their dataset are in fact captions from a publicly available Flickr image description dataset. This did, however, introduce another inductive bias for the dataset, as many annotations (24.0% of premises and 11.1% of hypotheses, according to a parser) do not result in a sentence, but are merely a caption-style noun-phrase. To provide more reliable data, a subset of the data (containing the entire test-set) was further verified by four other workers judging if an alleged entailment holds in their view. In 58% of cases, all five workers agreed, and in 98% at least three persons agreed, providing a gold label for the test-set.

MultiNLI Because the SNLI corpus only contains data of the very limited domain of image captions and the best model-performance is already almost human-level, Williams et al. [7] created another dataset for textual entailment in 2017. Having 433k examples, the corpus includes a wide range in the form of ten distinct genres of written and spoken English sentence in varying degrees of formality and styles. This does not only make the challenge harder, but also provides for evaluation on cross-genre domain adaption, as only half of the domains are included in the training set.

The data was collected in a similar way as SNLI and, like in SNLI test- and validation sets are validated by other human annotaters, whilst producing much longer sentences than the former. According to the authors, hypotheses tend to rely heavily on world knowledge, and are usually not of a different sentence structure than their premises, suggesting that alignment-based NLI models are unlikely to succeed.

A baseline for the dataset was created, following the technique of Mou et al. (2006), which achieved 72.4% in their matched- and 71.9% on the unmatched condition. A bi-LSTM that achieves an accuracy on 81.5% in SNLI achieved only 67.5%/67.1% in their respective conditions, indicating that the MultiNLI is a harder dataset to solve than SNLI.

Word Embeddings Explaining the details of word embeddings lies outside the scope of this paper . It is however worth mentioning how the different archi-

[5] Thus, when evaluating old methods on SNLI one has to keep in mind that baseline accuracy is 66.6% instead of 50%, as the latter two are merged to the category *non-entailment*.

tectures from the next sections incorporate the respective techniques. First off, there is no clear agreement on what technique rely on: while for example [8] and [16] rely on GloVe vectors, [4] and [5] (among others) rely on word2vec vectors.

Fine-tuning the word representations as part of the optimization progress may increase performance ([15] for example report a performance gain of 0.9 percentage points on SNLI when doing so), however it increases the number of parameters by several orders of magnitude (by doing so, [8] increase the number of parameters from 221k to about 10 million), which is why most approaches do not do so.

Another issue is how the respective approaches handle words for which no embeddings exist. According to [16], the SNLI corpus contains 37k unique tokens, 12.1k of which cannot be found in word2vec and 4.1k do not exist in GloVe. Most approaches assign random vectors, and fine-tune this subset throughout the training process (as [4], who increased the number of optimizable parameters from 111k to 3.8 million). Another interesting approach is used in [16]: instead of random embeddings for unknown words, they take the average of the embeddings of the 9 words around the unknown one, without updating these embeddings when training the model, saving millions of parameters in the process.

Sentence Encodings While it is easy to incorporate pretrained word embeddings, the same cannot be done for the corresponding sentence encodings. While an easy approach is to simply sum all the vectors of the word embeddings (termed continuous bag of words), doing so loses a lot of information.

Because of this, sentence encodings are a big part of the problem of recognizing textual entailment, and solved in many different ways. Due to their strength to handle sequences, Recurrent Neural Networks (RNNs) are a popular choice. More precisely, many approaches rely on Long Short-Term Memory (LSTM) RNNs to process sequential data. By incorporating a number of gate vectors to control the passing of information along the sequence, they allow for the modeling of long-range dependencies, where a state can incorporate information from possibly all states before it, such that at each word-position k, there is a hidden state h_k depending on the input word w_k and the content of the memory cell c_k. A sentence encoding could thus be modeled as the hidden state or the memory of the network at the final word.

To also consider future words for the representation of an input word, sometimes a bidirectional LSTM (biLSTM) is used instead.

The pool of ANNs for recognizing textual entailment can be subdivided into two groups. The first such group are *sentence-encoding based models*. In these architectures, both premise and hypothesis are first encoded into a high-dimensional space, before the embeddings of the two sentences are matched via another neural network. The second group of networks is that of the *models that use cross-features*. Models of this kind interweave the two aforementioned steps, often leading to a better overall result as the bottleneck of the general sentence representation is alleviated.

3.2 Sentence-Encoding Based Models

Sentence-encoding based models typically use a *Siamese architecture* – parameter-tied neural networks that are applied to encode premise and hypothesis. The building blocks for all sentence-encoding neural networks are the same – first off, the words are embedded in a high-dimensional space using *word2vec* or *GloVe*. Afterwards, a sentence encoding is calculated based on the word embeddings. The sentence encodings of the two sentences are then combined, before a neural network classifier is applied, matching the sentences to decide about the relationship between them, typically decided by a softmax over the three choices. Thus, this architecture consists of two steps: modeling the sentences and afterwards matching them. Figure 2 depicts this pipeline graphically.

While the choice of word-embeddings is only two-fold, different neural networks have been utilized as sentence encoders, such as LSTMs, CNNs, BiLSTMs or others. A disadvantage of such approaches is that they exclude models that compare the two input sentences at the word or phrase level.

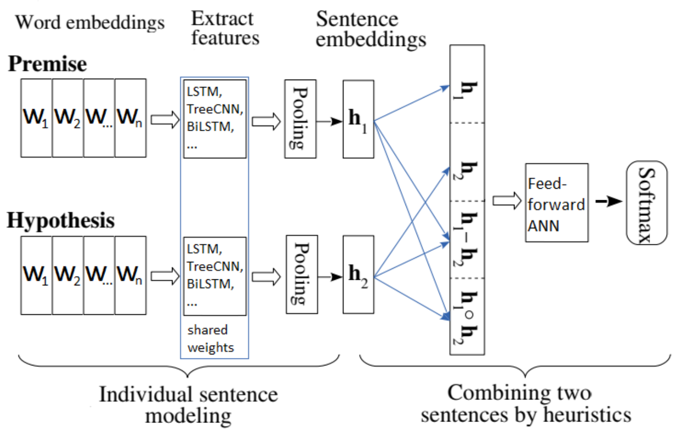

Fig. 2. General pipeline for sentence-encoding neural architectures. Adapted from [6].

To provide a baseline for the SNLI-dataset, [8] used 300-dimensional GloVe-embeddings, scaled down to 100 dimensions using a tanh-layer. After creating a sentence representation from these, Bowman et al. stacked the input layers, using it as input to three 200D-tanh-layers, concluding into a three-way softmax layer, all trained jointly with the sentence embedding model itself (and also fine-tuning word embeddings). In one of their runs, they calculated the sum of the respective words as sentence representation method, leading to an accuracy of 75.3%. Using an LSTM in another iteration provided an accuracy of 77.6% on the test set, on par with their lexicalized classifier.

Tree-Based Convolution and Matching Heuristics In their 2006 paper, Mou et al. [6] suggest some improvements to the above mentioned approach.

Firstly, they replace the sentence-embedding LSTM with a tree-based convolutional neural network (termed *TBCNN*). According to the authors, such a network fully uses syntactical information, countering the semantic mistakes at the phrasal level earlier approaches suffered under. To do so, a set of subtree feature detectors slides over the dependency tree, recognizing relations of individual components. Furhter advantages of CNNs in contrast to LSTMs are that they require less parameters, and that it uses also following words as context (similarly to biLSTMs)

Another addition of [6] is to leverage additional matching heuristics. In [8], the sentence encodings were simply concatenated to form the input of the matching network. Additionally to that, [6] use the element-wise product and element-wise difference of the output of the respective sentence encoders. While further heuristics (eg. euclidean distance, cosine measure, or word-by-word similarity matrices) might lead to even better results, these would have a runtime of $\mathcal{O}(\mathrm{n}^2)$ (n being the number of dimensions for the embedding), slowing down the matching process too much. These additions increased the accuracy on SNLI to 82.1%. Their latter addition was from then on incorporated for new baseline neural networks, for example the baseline performance of MultiNLI by [7].

Liu et al. [5] used a bidirectional LSTM to create the sentence representations, as it incorporates future context in processing a sequence, while not suffering from potential loss of word order information as CNNs tend to do. On top of that, they utilized an average-pooling layer. Inner attention is supposed to make the network utilize important words more than unimportant ones, leveraging past experience. The attention-weighted sentence representation is thus composed of a softmax over the output of an additional ANN layer coding the importance of the respective words – this increases the effect nouns, verbs and adjectives have in the overall sentence representation.

State of the Art The currently best performing sentence-encoding ANN outperforms Liu et al's model by another 1.8pps, achieving an accuracy of 86.1% on the SNLI corpus. Being published as recently as November 2017, the paper benefits from improvements made in other realms of Machine Learning, as for example the shortcut connections introduced in the *DenseNet* image recognition architecture[6], or the use of rectified linear units ($ReLU$).

The approach of Nie et al. [15] follows the aforementioned siamese architecture, first encoding each sentence in a high-dimensional feature space, to then match these sentence embeddings with a neural network classifier. As sequential sentence encoders, authors use multiple stacked bidirectional LSTMs with shortcut connections. This encoder does not require any syntactic information of the sentence, as TBCNNs do.

The authors stack three biLSTM layers, such that the input sequence for every biLSTM layer consists of the output of the layer below plus the original word embeddings (hence shortcut connections). Assuming i biLSTM layers and input sequence $w_1, w_2, \dots, w_n$, the input to a layer is thus:

[6] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. "Densely connected convolutional networks" (2016).

$$x_t^i = [w_t, h_t^{i-1}, h_t^{i-2}, \dots, h_t^1] \text{ for } i > 1, \text{ and } x_t^1 = w_t.$$

The output of a biLSTM layer follows as:

$$h_t^i = bilstm^i(x_t^i, t), \forall t \in [1, 2, \dots, n]].$$

(cf. [15]). To get the final vector representation of a sentence, a row-max-pooling is applied over the output of the final biLSTM layer, thus (with m layers)

$$v = max(h_1^m, h_1^m, ..., h_n^m) \in \mathbb{R}^{2d_m}.$$

While the sentence representation differs from the aforementioned approaches, the entailment classifier (second step of Figure 2) utilizes previous insights. The input to the matching network consists of the concatenation of the sentence embeddings, their element-wise distance and their element-wise product $m = [v_p, v_h, |v_p - v_h|, v_p \otimes v_h]$.

In detail, the author used the ADAM-optimizer for the cross-entropy with 1600 hidden units in the MLP classifier, incorporating dropout in the final layer. Word embeddings are 300-dimensional pretrained GloVe-vectors (fine-tuned for unknown words, giving an acccuracy gain of 0.9 pps). For the sentence embeddings, they used three layers of biLSTMs with 2048 dimensions for the final sentence representation. The authors used ablation studies to verify the effectiveness of the fine-tuning of the word-embeddings, the shortcut-connections (2 pps accuracy increase), and the number of biLSTM layers.

Using this setting, the authors achieved a performance of 86.1% on the SNLI-dataset as well as 74.6% in the Multi-NLI matched condition and 73.6% in the mismatched condition. This introduces a new state-of-the-art on both datasets for the sentence-encoding condition.

3.3 Cross-sentence Attention Mechanism

While the sentence-encoding architectures first generate independent sentence representations, it seems obvious that it can be useful to directly model the relation between premise and hypothesis, interweaving the sentence modeling and matching steps. To do so, many neural network models work by a matching-encoding approach, or by providing a cross-sentence attention mechanism that uses context-aware cross-features between the two sentences to encode them.

When deciding over NLI, it is often the case that for a hypothesis to contradict a premise, a single word or phrase (like the subject of the two sentences) mismatch may be sufficient to correctly identify a contradiction, whereas other word matching results are less important. This intuition is hard to be captured if we directly match two sentence embeddings.

While cross-sentence attention implementations generally provide a better performance (the currently best non-ensemble method provides an accuracy of 88.6% in contrast to 86.1% on the SNLI-corpus), they are computationally more intensive: The aforementioned sentence-encoding based models have a complexity of $\mathcal{O}(1)$, whereas a word-by-word attention mechanism increases this complexity to the order of $\mathcal{O}(\mathrm{n}^2)$ (cf. [6])

Incorporating Neural Attention A relatively simple architecture to incorporate attention into a neural RTE architecture was proposed by Rocktäschel et al. [4]. A simple way to transmit information from one LSTM to another is, to use the final cell state of the first LSTM as input for the second one. As this finite state is however a serious bottleneck, the idea of their mechanism is to use a neural attention component to find a soft-alignment between the two sentences. Thanks to this alignment, the LSTM does not need to capture the entire semantics of the sentence in one state.

In their framework, premise and hypothesis are subsequently analyzed by two different LSTMs (unlike the siamese architecture), where the one analyzing the hypothesis is initialized with the final cell state of the premise-analyzer. The final sentence-pair representation used for classification was then obtained from the *attention-weighted representation* of the premise and the final output-vector of the hypothesis. Note that while this architecture does incorporate an attention mechanism, it can still be considered as a sentence-encoding model, as it again reduces the two sentences to a single embedding vector before classifying a combination of these encodings.

The main idea of the word-by-word attention model is to introduce a series of attention-weighted combinations of the hidden states of the premise, where each weighted version of the premise is for a particular word in the hypothesis, called *attention vectors*. The second LSTM can thus attend over the first LSTM's output vectors, while processing the hypothesis one word at a time.

For each word w_j^{hy} of the hypothesis (of length N), the attention vector $\boldsymbol{a}_j$ is calculated as follows (with $\boldsymbol{h}_i^{pr}$ denoting the resulting hidden state corresponding to input word w_i^{pr} and M the length of the premise): $\boldsymbol{a}_j = \sum_{i=1}^{M} \alpha_{ij} \boldsymbol{h}_i^{pr}$.
The respective attention weight α_{ij} encodes the degree to which w_i^{pr} is matched to w_j^{hy}, and trained throughout the learning of the neural network (for more details, see [16] or [4]). As it is calculated using a softmax over all respective values from the premise's words, it holds that $\sum_j \alpha_{ij} = 1$.

This attention-weighted premise $\boldsymbol{a}_j$ models the relevant parts from the premise with respect to the j^{th} word of the hypothesis, and is used for the classification step of Rocktäschel et al's model [4]: the final sentence-pair classification uses as representation of the hypothesis the vector $\boldsymbol{h}_N^{hy}$ (the final state of the respective LSTM), and as representation for the premise $\boldsymbol{h}_N^a$, with $\boldsymbol{h}^a$ defined as:

$$\boldsymbol{h}_k^a = \boldsymbol{a}_k + \tanh(\boldsymbol{V}^\alpha \boldsymbol{h}_{k-1}^\alpha).$$

to predict if the pair constitutes a valid entailment. Thus, each word of the hypothesis is matched to an attention-weighted sum over hidden states of the premise. As the second LSTM weights over all output vectors of the premise for every word in the hypothesis, the computational complexity grows to $\mathcal{O}(n^2)$.

In their analysis, the authors show that a network that uses neither attention nor the Siamese architecture from the previous section performs worse than its Siamese counterpart (both achieving 80.9% on SNLI, the former however with 116 in contrast to 100 dimensions). Using an attention-mechanism increases the accuracy to 82.3%, and the explained word-by-word attention even to 83.5% (keep in mind that this implementation uses a unidirectional LSTM and none of the aforementioned additions from the previous sections as the matching heuristics). The authors thus provide evidence that RNNs over a single chain of two sentences can be more informative than two separate, weight-sharing RNNs, even when using a similar number of parameters.

The authors argue that word-by-word attention is better due to the model being able to check for entailment or contradiction of individual words in the hypothesis. By visualizing the word-by-word attention weights for some sample sentences, they try to provide evidence supporting that claim. To give an example, consider this sentence pair:

Premise: A young girl kisses a ring-bearing boy at a wedding.
Hypothesis: Two kids are playing tag.

The attentive system yields the highest activation for *girl* and *boy* when processing the word *kids* from the hypothesis. This may provide evidence that it correctly identifies the one-to-many mapping of the former words to the latter.

Using a matching-LSTM An approach by [16] builds on top of this attention-model, but uses a single LSTM that performs word-by-word matching of the hypothesis with the premise. The idea behind not using separate sentence-encodings that are subsequently matched is to be able to give word-level matching results, instead of having to shrink sentences into a fixed-size vector. To do so, the authors introduce a *match-LSTM* which performs a word-by-word matching of the hypothesis with the premise, allowing to remember critical matching results while attaching little importance to less important matchings.

The architecture suggested by the authors consist in the first step of two independent LSTMs on premise and hypothesis. Afterwards, they generate attention vectors, matching premise-tokens to hypothesis-tokens based on the hidden states of the previous LSTMs. While the process works similarly to [4], it incorporates the hidden state of their match-LSTM, modeling the matching between premise and hypothesis. Using an LSTM to do so allows to remember essential matches, while non-important ones are forgotten.

The final classificator is another LSTM, taking at each position both $\boldsymbol{a}_j$ (the attention-weighted version of the premise for the j$^{\text{th}}$ word in the hypothesis) and $\boldsymbol{h}_j^{hy}$ (the hidden state for the j$^{\text{th}}$ word) as input (cf. [16] for a more detailed description). The final state of this match-LSTM ($\boldsymbol{h}_N^m$) is then used to predict the final label *entailment, neutral* or *contradiction.*

This implementation achieved a performance of 86.1% on the SNLI dataset, on par with the work of [15], but significantly outperforming previous approaches at the time of publication.

3.4 Other Neural Approaches

There are a number of other neural approaches for solving textual entailment that could be mentioned. While many of them either rely on the sentence-encoding based model or the attentive model, they introduce new ideas and concepts, or rely on other network architectures.

The current state-of-the-art on the SNLI dataset (for non-ensemble methods) is 88.8%, mutually held by a Neural Semantic Encoder as well as a Knowledge-based Inference Model (KIM)[7]. While the former presents a highly flexible new type of neural network for a range of NLP tasks, the latter tries to enrich inference models by external knowledge, extracting data about hyponomy, synonymy, etc. from the WordNet-database.

3.5 Discussion of Neural Architectures

ANNs provide a huge quality gain in detecting natural language inference. While during the times of the PASCAL RTE challenges performances ranged from 50% to 70%, the most simple ANN architecture already provides an accuracy of 77% on the far more complex SNLI dataset, with current methods reaching almost 90%. In fact, the performance is so high, that the question arises if more performance is even possible – as mentioned in section 3.1, a five-person human agreement came about in only 58% of the dataset. There is reason in arguing to consider the SNLI-dataset as solved, and drop is as benchmark dataset in favor of the more complex MultiNLI-dataset.

Another question is however, how much of the continuous progress in solving such datasets[8] is due to better models, and how much else comes from other factors, like newer models using higher-dimensional embeddings, or the fact that now mostly the tanh-activation-function is dropped in favor of the ReLU-function (leading to a performance gain of 0.9 percentage points for [15]).

It is interesting that the performance is so good, considering almost all world knowledge they have (next to the pre-trained word embeddings) comes from the datasets they train on alone. As [8] also stated, many of commonly mis-interpreted entailment pairs rely on context-specific information and world knowledge. It will be interesting to see how much performance-gain will be achieved, once such a world knowledge can be reliably incorporated into neural architectures, the KIM model being a pioneer on that field.

4 Conclusion

To conclude, it is obvious that through the use of neural networks and word embeddings, textual entailment recognizers have gone through their newest step of evolution and they have come a long way from the early text-based systems.

While at the first RTE challenge in 2005 no system reached a performance of more than 57%, current systems arguably reach human gold standard already. With such a progress, it will be interesting to see where research will lead in the future. One possibility is definitely a look back at the original systems, and finding a way of incorporating word knowledge and world knowledge (as e.g. WordNet).

The current and coming systems will not only have a diverse range of possible applications, as has been shown, but their application fields are also going

[7] Chen, Q., Zhu, X., Ling, Z.-H., Inkpen, D., and Wei, S. (2017). Natural Language Inference With External Knowledge. Retr. from https://arxiv.org/pdf/1711.04289.pdf

[8] See https://nlp.stanford.edu/projects/snli/ for an overview of the current state-of-the-art and the models that reached it.

through constant improvements. QA systems or dialog managers are a good example of current hot topics which have already in part been shown to profit from incorporating NLI systems, such that seeing textual entailment systems being used in a variety of these applications is definitely the next logical step.

References

1. Ion Androutsopoulos and Prodromos Malakasiotis. “A Survey of Paraphrasing and Textual Entailment Methods”. In *Journal of Artificial Intelligence Research* (2009).
2. Bill MacCartney and Christopher D Manning. “Natural logic for textual inference”. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing* (2007), pp. 193–200.
3. Bill MacCartney. “Natural language inference”. Ph.D. dissertation, Stanford University (2009).
4. Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. “Reasoning about entailment with neural attention”. In *Proceedings of the International Conference on Learning Representations* (2016).
5. Liu, Y., Sun, C., Lin, L., and Wang, X. “Learning Natural Language Inference using Bidirectional LSTM model and Inner-Attention” (2016).
6. Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. “Natural language inference by tree-based convolution and heuristic matching”. In *Proc. of ACL* (2016).
7. Williams, A., Nangia, N., and Bowman, S. R. “A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference” (2017).
8. Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. “A large annotated corpus for learning natural language inference”. In *Proc. of the 2015 Conference on Empirical Methods in NLP* (2015).
9. Cleo Condoravdi, Dick Crouch, Valeria de Paiva, Reinhard Stolle, and Daniel G. Bobrow. “Entailment, intensionality and text understanding”. In *Proceedings of the HLT-NAACL 2003 Workshop on Text Meaning* (2003).
10. Ido Dagan, Oren Glickman, and Bernardo Magnini. “The PASCAL recognising textual entailment challenge”. In *Machine learning challenges. Evaluating predictive uncertainty, visual object classification, and recognising tectual entailment* (2005).
11. Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Bryan Rink, and Ying Shi. “Recognizing textual entailment with LCC’s GROUNDHOG system”. In *Proceedings of the Second PASCAL Challenges Workshop*, vol. 18 (2006).
12. S. Harabagiu and A. Hickl. “Methods for using textual entailment in open-domain question answering”. In *Association for Computational Linguistics (ACL)* (2006).
13. Bos, Johan, and Katja Markert. “Combining shallow and deep NLP methods for recognizing textual entailment”. *In Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment* (2005), pp. 65–68.
14. Malakasiotis, Prodromos, and Ion Androutsopoulos. “Learning Textual Entailment using SVMs and String Similarity Measures”. In *Proceedings of the ACLPASCAL Workshop on Textual Entailment and Paraphrasing* (2007), pp. 42–47.
15. Yixin Nie and Mohit Bansal “Shortcut-stacked sentence encoders for multi-domain inference”. In *Proceedings of The Second Workshop on Evaluating Vector Space Representations for NLP* (2017).
16. Shuohang Wang and Jing Jiang. “Learning natural language inference with LSTM”. In *NAACL* (2016).
17. Zaenen, A., Karttunen, L., and Crouch, R. “Local textual inference: Can it be defined or circumscribed?”. *In Proc. of the ACL workshop on Empirical Modeling of Semantic Equivalence and Entailment* (2005), pp. 31–36.