LEAN: A Lexical Entailment Augmented Network for Natural Language Inference

Aurélien Coet

University of Geneva
Faculty of Sciences,
Dept. of Computer Science

June 2019

UNIVERSITÉ
DE GENÈVE
FACULTÉ DES SCIENCES

Contents

List of Figures

List of Tables

List of Acronyms

AI *Artificial Intelligence*. 1

BERT *Bidirectional Encoder Representations from Transformers*. iv, 23, 55

bi-LSTM *bidirectional LSTM*. iv, 12–14, 17, 18, 31, 33

biLM *bidirectional Language Model*. 22

BiMPM *Bilateral Multi-Perspective Matching*. 18

c-LSTMs *coupled-LSTMs*. 16

CAFE *Comprop Alignment-Factorised Encoders*. 18

CDSMs *Compositional Distributional Semantic Models*. 10

CNN *Convolutional Neural Network*. 13, 15

CoVe *Context Vectors*. iv, 20–22, 24

DIIN *Densely Interactive Inference Network*. 18

DiSAN *Directional Self Attention Network*. 15

DL *Deep Learning*. 4

DMAN *Discourse Marker Augmented Network*. 21, 24

DMP *Discourse Marker Prediction*. 21

DR-BiLSTM *Dependant Reading Bidirectional LSTM*. 18

DRCN *Densely-connected Recurrent and Co-attentive neural Network*. iv, 18, 19

DSA *Dynamic Self Attention*. 15

ELMo *Embeddings from Language Models*. 22–24

ESIM *Enhanced Sequential Inference Model*. i, iv, v, 8, 17–19, 22, 30–39, 41–46, 48–50

GPT *Generative Pre-training Transformer*. iv, 22–24

GRU *Gated Recurrent Unit*. 12

KIM *Knowledge-based Inference Model*. 19, 49

LE *Lexical Entailment*. iv, v, 4, 24–26, 28, 29, 34, 39, 40, 42–46, 48–52, 55

LEAN *Lexical Entailment Augmented Network*. iv, v, 4, 26, 39–55

LEAR *Lexical Entailment Attract-Repel*. iv, 28, 29, 40, 44, 45, 48–52

LMs *Language Models*. 21, 23

LSTM *Long Short Term Memory*. 7, 12, 16, 31, 32

MLM *Masked Language Model*. 23

MLP *Multi-Layer Perceptron*. 12, 18

mLSTM *match-LSTM*. 16

MT *Machine Translation*. 20, 21

MultiNLI *Multi-Genre Natural Language Inference*. v, 7–12, 14, 15, 17, 19, 20, 23, 24, 34, 36–38, 47–50, 52, 55

NLI *Natural Language Inference*. iv, 1–5, 7–9, 11–13, 16–23, 30, 31, 34–37, 39, 45, 49, 54, 55

NLP *Natural Language Processing*. 1, 3, 14, 20–22

NLU *Natural Language Understanding*. 1–4, 11, 23

NNs *Neural Networks*. 4

OANC *Open American National Corpus*. 7

ReSAN *Reinforced Self Attention Network*. 15

rLSTM *re-read LSTM*. 16

RNN *Recurrent Neural Network*. 12, 13, 18, 35

RTE *Recognising Textual Entailment*. 1, 2, 9, 10, 12, 21

SDSN *Supervised Directional Similarity Network*. iv, 29, 40, 41, 45, 51, 52

SICK *Sentences Involving Compositional Knowledge*. 7, 10

SNLI *Stanford Natural Language Inference*. v, vi, 5–12, 14, 15, 17, 19–21, 23, 24, 34, 36–38, 46, 48–53, 55

SPINN *Stack-augmented Parser-Interpreter Neural Network*. 13, 14

SRL *Semantic Role Labelling*. 19

TBCNN *Tree-based Convolutional Neural Network*. 13, 14

TreeRNNs *Tree-structured Recursive Neural Networks*. 13, 14

XNLI *Cross-lingual Natural Language Inference*. 10

Chapter 1

Introduction

1.1 Definition and Scope of NLI

Natural Language Understanding (NLU) is a sub-domain of *Natural Language Processing* (NLP) and *Artificial Intelligence* (AI) which aims at giving computers the ability to understand and interpret human languages. In order to fulfill that goal, a number of difficult tasks involving syntactic and semantic aspects of natural language have been devised, such as *text summarisation*, *sentiment analysis* or *relation extraction*. Among these tasks is the central problem of *Natural Language Inference* (NLI), also known as *Recognising Textual Entailment* (RTE) (Dagan et al., 2006).

In NLI, the objective is to recognise whether a sentence p, called *premise*, entails another sentence h, called *hypothesis*. Here, we define *entailment* as the relation of information inclusion between the premise and hypothesis. That is, a sentence p entails h if and only if all of the information given in h is also present in p. In other words, a hypothesis is entailed by a premise if it can be inferred from it. An example is given below:

p: Two boys are playing football in a grass field.
h: Children are playing outside.

For a human reader, recognising that p entails h in the example above is done easily. Our brain is naturally able to determine that *two boys* are *children*, that a *grass field* is *outside*, and that *playing football* is summarised by *playing*. However, allowing a computer to perform the same task proves to be particularly hard. The difficulty of NLI comes not only from the complex nature and ambiguity of natural language, but also from the fact that the decision whether a sentence entails another is based on human judgement and not some kind of formal logic. Indeed, in NLI the relation linking a premise and hypothesis is usually chosen to be what someone would decide upon reading the sentences. Hence, giving a system the ability to detect entailment not only involves developing an understanding of the structure and meaning of language, but also reproducing the line of thought of humans.

If we consider the steps through which our brain went to recognise entailment in the previous example, a number of complex tasks both on the syntactic and semantic level can be identified. First, before the sub-parts of the two sentences can be compared, they must be identified and matched. It is hence necessary to parse the sentences into their subject, verb, object and complements, which implies having some kind of knowledge of English grammar. Once this is done, the semantic relations that lie between the matched phrases then need to be recognised and composed to make a decision about the sentence pair being observed. While entailment can easily be inferred from some of those relations (e.g. *playing football* obviously entails *playing*), it can also be very hard to infer from others (some advanced knowledge of the world is necessary to be able to determine that a *grass field* is located *outside*). Of course, the steps described above aren't always necessary for a system to perform well on NLI, but they illustrate the kind of challenges being faced when working on the problem.

So far, we have only defined NLI as a binary classification task in which a choice has to be made between entailment/no entailment based on a premise-hypothesis pair. In most formulations of NLI problems, however, the actual objective isn't to distinguish between two but three different classes: *entailment*, *contradiction* and *neutral* (S. Bowman et al., 2015; Conneau, Rinott, et al., 2018; Giampiccolo et al., 2008; Marelli et al., 2014; Williams et al., 2018). This particular kind of formulation makes the task at hand even harder, as not only information *inclusion*, but also information *exclusion* need to be recognised.

Sentence pair	Relation
p: A woman having a beverage. **h:** Woman has a drink.	**entailment**
p: Men are sitting at a table. **h:** People standing near a table.	**contradiction**
p: A man and his dog playing frisbee. **h:** A man is having fun with his dog.	**neutral**

Table 1.1: Examples of sentence pairs and their associated labels

Let's consider the three examples of sentence pairs and their associated labels in table 1.1. In order to properly classify them, a system must be able to identify that *beverage* and *drink* are synonyms in the first pair, that *standing* and *sitting* are antonyms in the second, and finally that *playing frisbee* doesn't necessarily imply *having fun* but doesn't contradict it either. Furthermore, while the sentences presented here do not illustrate it, premises and hypotheses in NLI tasks can sometimes contain grammatical errors and spelling mistakes, as they are usually written by humans. This kind of errors shouldn't prevent a system from correctly classifying instances, which only adds to the difficulty of the problem.

In their *MultiNLI* paper (Williams et al., 2018), the authors argue that the main difficulty in RTE is to extract meaningful representations for the sentences that compose the premises and hypotheses of an NLI data set, which makes the task particularly interesting for representation and transfer learning. They also underline how the large variety of linguistic phenomena that must be handled by models to recognise entailment makes it a good benchmark on NLU.

1.2 Interest and Applications

Considering all the elements mentioned in the previous section, one can easily imagine why natural language inference is such an important research interest in the field of NLP. The complexity of the task and its semantic relevance make it an essential aspect of NLU and a major problem that needs to be tackled in this area. The interest of NLI however goes beyond academic research, and numerous applications would benefit from the ability to perform inference on human language:

- In *automatic text summarisation* (Das and Martins, 2007), the objective is to automatically produce a summary for a piece of text or a document. The result should be short and remove any kind of redundancy from its source, but without losing any of the important information it contains.

  In such a situation, NLI can for example be used to detect if any sentence of the summary can be inferred from the others (which would mean that it is redundant) (Dagan et al., 2006), or to verify that the summary is well entailed by the original text from which it was generated (to ensure that it doesn't include any additional or unrelated information) (Pasunuru et al., 2017). Inference can also be used to directly address the task at hand, by determining which sentences in the original document are entailed by other larger chunks of text and extracting them to build a summary.

- *Opinion summarisation* (Condori and Pardo, 2017) is a task that has seen a significant surge in interest over the past years, mainly due to the fast growth of social networks and online shopping websites. The idea in opinion summarisation is to analyse pieces of text written by different people on a specific subject (such as product reviews on Amazon or political opinions on Twitter) and to extract the general sentiments that are shared by multiple persons.

  In this case, NLI can be used in a similar fashion as for automatic text summarisation: a sentence or a piece of text that is entailed by multiple opinions can be considered to summarise them well, as it contains no contradictory or additional information.

- In *reading comprehension* (Hirschman et al., 1999), a system takes some document as input and answers questions about it by searching for relevant information in the text.

  In this type of problem, NLI can be used to find the sentences in the source text that can be inferred from the questions and use them to build answers. It can also serve to choose the best answer between potential solutions by determining which ones can be inferred from the questions with more confidence.

- *Question answering* is a task very similar to that of reading comprehension where the objective is to answer open domain questions by using diverse sources of information. In that situation, NLI can serve similar purposes as in reading comprehension.

1.3 Contents of this Document

Thanks to the relatively recent creation of large scale natural language inference data sets (more on that in chapter 2) and the fast development of *Neural Networks* (NNs) over the past few years, the NLU community has come up with numerous *Deep Learning* (DL) models to address the problem of NLI. However, as we will see in the next chapter, only very little work has been done on investigating the use of lexical level information to help DL models recognise entailment.

In this work, we propose to implement a well-known model for NLI and to augment it with lexical entailment information, in order to assess whether this helps it on the task of recognising textual entailment.
Our implementation of the original model is first trained and tested on three famous corpora for NLI, and its classification accuracy on them is defined as the baseline to beat.
We then investigate multiple ways to include existing metrics designed to detect entailment between words in the model, and we measure whether this additional information improves its performance. The experiments we conduct show that our model enhanced with lexical information does indeed perform better on the task of NLI compared to our baseline (although only by a small margin), hinting at the fact that lexical entailment can help systems to better recognise inference at the sentence level, and opening up new avenues for research in this area.

The rest of this document is organised as follows:

- In chapter 2, we provide a complete literature review on the topics addressed in this work. The chapter is sub-divided in two main sections that focus on natural language inference and lexical entailment, respectively.
- In chapter 3, we describe the model we used as a baseline in this work in detail, and we present our implementation for it with *PyTorch*[1], a popular deep learning framework. We also provide information about the specific parameters we used to train the model, and we report its performance on three famous data sets for NLI.
- In chapter 4, we present the *Lexical Entailment Augmented Network* (LEAN), a model built on top of the baseline presented in chapter 3 which uses additional lexical entailment information to perform inference. Different ways of including lexical entailment and different LE metrics are explored, and their influence on the model's performance is reported.
- In chapter 5, the results obtained with LEAN are compared with the baseline, and they are analysed and discussed in depth. A small ablation analysis is also proposed on the best performing version of LEAN to determine which aspects of the lexical entailment information participate the most to its performance.
- Finally, chapter 6 proposes a conclusion for this work and opens up new directions for further research in the area of natural language inference.

[1] `https://pytorch.org/`

Chapter 2

Related Works

2.1 Natural Language Inference

In the first section of this chapter, we investigate previous work in the domain of NLI. We focus in particular on models that have addressed the problem with approaches in deep learning, and we present the specific tasks and data sets that were used to train them.

2.1.1 NLI Tasks and Data Sets

Over the years, a number of tasks and data sets have been devised for the problem of natural language inference. In particular, the relatively recent introduction of large scale, high quality corpora enabled the development of many deep learning models for NLI. In this sub-section, we describe the most prominent ones and compare them.

2.1.1.1 The SNLI Corpus

In 2015, Bowman et al. presented the *Stanford Natural Language Inference* (SNLI) corpus (S. Bowman et al., 2015), a large scale, manually generated and annotated data set of sentence pairs labelled for textual entailment. With a total of 570,152 instances, the corpus was the first of its kind, and its impressive size sparked the apparition of numerous deep learning models for natural language inference.

The SNLI corpus is composed of pairs of sentences called *premises* and *hypotheses* and labelled with one of the three classes *entailment*, *contradiction* and *neutral*. The data is divided into a training set containing 550,152 sentence pairs and a development and a test set containing 10,000 pairs each. The objective in SNLI is to correctly predict the label associated to each instance in its test set.

To build SNLI, the authors used the Amazon Mechanical Turk[1] crowd-sourcing platform. There, human workers were presented with series of premises and asked

[1] `https://www.mturk.com/`

to write three hypotheses for each of them: one that was entailed by the premise (labelled with *entailment*), one that contradicted it (labelled with *contradiction*), and one that wasn't entailed by nor contradicted it (labelled with *neutral*). Specific indications were given to the workers to guide them in their task (advices on sentence length, complexity, etc.), as well as restrictions (it was for example forbidden to reuse the same sentence twice). Examples of sentence pairs and their associated labels are presented in table 2.1 (taken directly from the original paper).

Premise	**Hypothesis**	**Label**
A soccer game with multiple males playing.	Some men are playing a sport.	entailment
A man inspects the uniform of a figure in some East Asian country.	The man is sleeping	contradiction
An older and younger man smiling.	Two men are smiling and laughing at the cats playing on the floor.	neutral

Table 2.1: Examples of instances from the SNLI corpus (S. Bowman et al., 2015)

For the premises of SNLI, the authors extracted captions from the Flickr30k corpus (Young et al., 2014), a crowd-sourced data set composed of image descriptions. The motivation for using captions was that it helped solve the problem of *event* and *entity co-reference* in sentence pairs.
Event/entity co-reference refers to the situation were the premise and hypothesis in a pair mention some entity or event, but it cannot be trivially determined whether they are the same or not. In the SNLI paper (S. Bowman et al., 2015), the example of the sentences *"A boat sank in the Pacific ocean"* and *"A boat sank in the Atlantic ocean"* is given. In that situation, it is not clear whether it should be considered that the event being referred to is the same or not. If it is, the label associated to the pair should be *contradiction*, as the location of the boat in the hypothesis is different from the one in the premise. However, if the events are considered to be different, the associated label should be *neutral*, because the sentences don't contradict each other since they refer to different boats and accidents.
Using captions helped solve this problem, as all events or entities in the premises belonged to some image on which the hypotheses written by the workers were supposed to be based too. This meant that events and entities mentioned in the hypotheses could always be assumed to be the same as those in the premises when labelling pairs.

Once data collection was complete, an additional validation round was applied on about 10% of the corpus. Instances from the data set were presented to the workers without their associated class, and they were asked to label them. Four different persons validated each pair, and a majority vote was used to determine the gold label. Sentence pairs for which no agreement was reached were kept in the corpus but labelled with "-", to indicate that no gold label could be selected. The rate of agreement during this phase was of 98%, with unanimity obtained in about 58% of cases.

Aside from the corpus described above, Bowman et al. also proposed several

models trained and tested on the data in their paper. The two best performing ones were a lexicalised classifier and a simple *Long Short Term Memory* (LSTM) neural network. These were defined as the baselines to beat when the paper was published. Additionally, the LSTM was tested on the SICK corpus (an older, smaller scale NLI data set) with transfer learning. The authors first trained the model on SNLI and then fine-tuned it on SICK's training set. With this approach, they obtained new state-of-the-art results, which showed the potential of the SNLI data set for training efficient deep learning models on the task of recognising entailment.

It is generally accepted that the manual construction and annotation of the SNLI corpus by human workers make it a high quality resource. Its large size also allows it to be particularly appropriate for uses in modern deep learning approaches to NLI. However, the corpus has its limitations. Williams et al. (2018) explain in their MultiNLI paper that because all sentences in SNLI were built on a single type of textual resource (namely image captions), they do not allow for good generalisation on other kinds of texts and lack certain important linguistic phenomena (such as temporal reasoning or modality, among other examples). These were some of the reasons for the creation of the MultiNLI corpus.

2.1.1.2 The MultiNLI Corpus

The *Multi-Genre Natural Language Inference* (MultiNLI) corpus (Williams et al., 2018) was created at the University of New York in 2017 to remedy the shortfalls of SNLI. It consists in 432,702 pairs of sentences labelled for textual entailment and covers a wide range of textual styles and topics. The data is split in a training and development set that are available online[2], as well as test sets that can only be accessed through Kaggle competitions in unlabelled form[3][4].

MultiNLI is very similar to SNLI both in its structure and in the way it was constructed: the sentence pairs that compose it were produced through a crowd-sourcing effort that followed the same protocol, and they have the same form.

The main difference between the two corpora is the type of textual resources that were used to produce their premises. Compared to SNLI, many more types of text were used for MultiNLI. More specifically, the authors extracted premises from ten different types of sources written in English. Nine of them were part of the *Open American National Corpus* (OANC), which contains transcriptions of real world conversations, reports, speeches, letters, non-fiction works, articles from magazines, travel guides and short posts on linguistics for non-specialists. The tenth genre of text used in MultiNLI was fiction and contained a compilation of open access works written between 1912 and 2010, covering different styles such as science-fiction, mystery or adventure.

One of the problems that MultiNLI's authors identified in SNLI was that it was *"not sufficiently demanding to serve as an effective benchmark for NLU"* (Williams et al., 2018). Hence, in order to make MultiNLI more difficult, they split its data such

[2] `https://www.nyu.edu/projects/bowman/multinli/`
[3] `https://www.kaggle.com/c/multinli-matched-open-evaluation`
[4] `https://www.kaggle.com/c/multinli-mismatched-open-evaluation`

that only five genres of text were covered in its training set, and its testing set was divided in two categories: a *matched* set only containing premises extracted from the same genres as the training set, and a more challenging *mismatched* set containing premises from all ten genres selected during data collection. The insight was that the matched set would allow to *"explicitly evaluate models [...] on the quality of their sentence representations within the training domain"*, whereas the mismatched version would allow to test *"their ability to derive reasonable representations in unfamiliar domains"* (Williams et al., 2018).

To verify that their corpus did indeed improve the difficulty compared to SNLI, the authors of MultiNLI trained and tested several baselines on it, as well as the then state of the art ESIM model (Chen, Zhu, Ling, Wei, et al., 2017a). They wrote their own implementation of ESIM and tested it on SNLI, which yielded 86.7% accuracy (meaning that the model correctly classified 86.7% of the instances it saw in SNLI's test set). On MultiNLI, their implementation only performed at 72.4% accuracy on the matched set and 71.9% on the mismatched version, effectively proving that their new corpus represented a greater challenge than SNLI for natural language inference.

Table 2.2 below presents some examples of sentence pairs extracted from the MultiNLI corpus (taken from the data set's official website[5]).

Text type	**Premise**	**Hypothesis**	**Label**
Letters	Your gift is appreciated by each and every student who will benefit from your generosity.	Hundreds of students will benefit from your generosity.	neutral
Telephone	yes now you know if if everybody like in August when everybody's on vacation or something we can dress a little more casual or	August is a black out month for vacations in the company.	contradiction
9/11 report	At the other end of Pennsylvania Avenue, people began to line up for a White House tour.	People formed a line at the end of Pennsylvania Avenue.	entailment

Table 2.2: Examples of instances from the MultiNLI corpus (Williams et al., 2018)

While MultiNLI is now widely accepted as the de facto standard for training and evaluating NLI models, the corpus is not devoid of flaws. In particular, two papers published in 2018 (Gururangan et al., 2018; Poliak et al., 2018) showed that it was possible to predict the labels associated to sentence pairs in SNLI and MultiNLI with relatively high accuracy by solely looking at the hypotheses. The explanation given by the two works to explain this phenomenon was that the corpora suffered from annotation artifacts, such as specific sentence lengths or choices of words by the annotators for given classes.

[5] `https://www.nyu.edu/projects/bowman/multinli/`

2.1.1.3 The Breaking NLI Data Set

In 2018, Glockner et al. proposed a new benchmark to evaluate whether the models trained to perform NLI were efficient at solving problems that involve lexical inferences and world knowledge. The corpus, named the *Breaking NLI* data set (Glockner et al., 2018), consists only in a test set of 8,193 sentence pairs and is meant to be used to evaluate models previously trained on SNLI.

To build their data set, Glockner et al. extracted premises from SNLI's training set and applied automatic transformations on them to produce hypotheses where only a single word has been replaced. Replacement words were chosen to generate hypotheses that were either entailed by the selected premises or contradicted them (though neutral examples were also obtained in some cases as a by-product).
After the sentence pairs and their associated label were obtained with the automatic procedure, they were further validated by human annotators through a crowdsourced effort, to ensure their correctness. Examples of pairs and their labels are provided in table 2.3 (directly taken from the paper).

Premise	**Hypothesis**	**Label**
The man is holding a saxophone	The man is holding an electric guitar	contradiction
A little girl is very sad	A little girl is very unhappy	entailment
A couple drinking wine	A couple drinking champagne	neutral

Table 2.3: Examples of instances from the *Breaking NLI* data set (Glockner et al., 2018)

Once they had completed data collection and validation, the authors evaluated a number of models that performed well on SNLI and MultiNLI on their own test set. All models performed significantly worse on their data, except for one: KIM (Chen, Zhu, Ling, Inkpen, et al., 2018), which makes use of external lexical information to perform classification.

The results show that most models trained on the current best corpora for NLI have poorer generalisation capability than was previously thought, and that further improvements in the NLI task definition would be necessary to alleviate this problem.

2.1.1.4 Other Resources

In addition to the data sets mentioned in this sub-section, other resources exist for the task of recognising textual entailment. These won't be discussed in depth here, as we consider them of somewhat lesser relevance in the specific context of deep learning for NLI. Nevertheless, we list them below for the sake of completeness:

- The *Recognising Textual Entailment* (RTE) challenge benchmark (Dagan et al., 2006) was probably one of the first data sets to ever propose a unified task on which to evaluate NLI models. Presented during a workshop on textual entailment in 2005, the corpus is composed of sentence pairs produced by

human annotators and labelled with *entailment/no entailment*. It is split in a training and a test set containing 567 and 800 instances, respectively.

After the first edition in 2005, subsequent versions of RTE were proposed from 2006 to 2011 (RTE-2 to RTE-7), all with sizes similar to the first instance. Starting from 2008 (RTE-4), the challenge moved from a binary classification setup to a three class problem (*entailment/contradiction/unknown*).

Even though the manual definition of the RTE data sets makes them high quality resources, their limited size prevents them from being of any use in the training of models that need to learn representations on large quantities of data, such as deep neural networks.

- The *Sentences Involving Compositional Knowledge* (SICK) data set (Marelli et al., 2014) was presented during a workshop on semantic evaluation in 2014 as a benchmark for the evaluation of *Compositional Distributional Semantic Models* (CDSMs). The corpus consists in 10,000 sentence pairs labelled with a score on a 5-point scale for semantic relatedness and with *entailment/contradiction/neutral* for textual entailment.

  To generate the pairs, image captions from the 8K ImageFlickr[6] and SemEval 2012 STS MSRVideo Description data sets were used as premises, and automatic transformations were applied on them to produce hypotheses. The results were then manually labelled by human annotators and a majority vote was used to validate the procedure.

  Although SICK is larger than the RTE data sets by an order of magnitude, its size is still to small to be used efficiently in the training of deep learning models. In their SNLI paper, Bowman et al. also denote how the automatic aspect of the sentence pairs generation in SICK introduced *"some spurious patterns into the data"* (S. Bowman et al., 2015).

- The *denotation graph* (Young et al., 2014) consists in a large hierarchical set of sentences connected to each other through the relation of entailment.
  As with the SICK data set, Bowman et al. explain in their SNLI paper that the automatic generation of the denotation graph makes it too noisy to be usable in the training of data intensive models (S. Bowman et al., 2015).

- The *Cross-lingual Natural Language Inference* (XNLI) corpus (Conneau, Rinott, et al., 2018) is a data set that extends the development and test sets of the MultiNLI corpus with 7,500 human-annotated pairs of sentences in 15 different languages.

  While the pairs follow the exact same structure as those in the SNLI and MultiNLI data sets, the primary focus of the XNLI corpus is not on recognising textual entailment, but rather on providing a strong benchmark for the evaluation of systems on the task of *cross-lingual natural language understanding*.

[6] `http://nlp.cs.illinois.edu/HockenmaierGroup/8k-pictures.html`

2.1.2 Deep Learning Models for NLI

Since the release of the SNLI corpus in 2015, a wide array of deep learning models have been devised for the task of natural language inference. Those models can roughly be regrouped into three main categories: sentence vector-based models, sentence matching models and transfer learning approaches.

2.1.2.1 Sentence Vector-based Models

As mentioned in the MultiNLI paper (Williams et al., 2018), the wide variety of linguistic phenomena covered by natural language inference not only makes it a good benchmark for NLU, but also an excellent supervised task for the learning of *sentence embeddings*, vector representations that capture the semantics of sentences. For this reason, there exist a large number of models focused on learning general sentence representations from NLI in the literature. These models are often referred to as *sentence vector-based*.

The general architecture of sentence vector-based models consists in two main components: a *sentence encoder* (or *sentence model*) and a *classifier* (or *matching layer*). The task of the sentence encoder is to learn generic representations for the premises and hypotheses in some NLI problem, and the classifier then has to somehow combine these representations to predict the relationship that exists between the two sentences. This structure is often referred to as the *Siamese architecture* (Bromley et al., 1994), represented in figure 2.1. Note that the two sentence encoders in the image share the same weights (the same network is applied on both the premise and hypothesis).

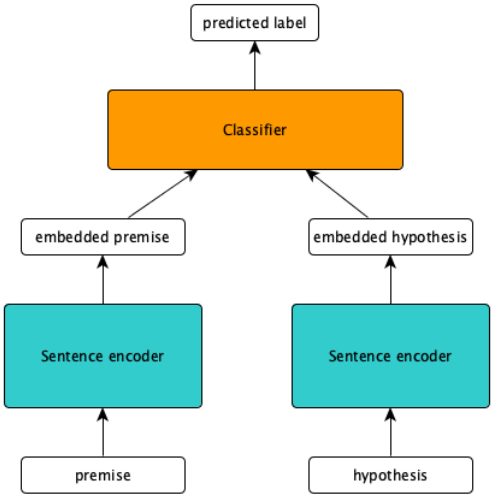

Figure 2.1: Representation of the Siamese architecture for NLI

Most sentence vector-based model use the same type of classifier to perform predictions on NLI. The architecture of such a classifier, represented in figure 2.2, was first introduced in a paper by Mou et al. (2015) and has since often been reused almost as is in many other sentence vector-based models. In the figure, h_1 and h_2 are vector representations learned by a model's sentence encoder for the premise and

hypothesis in some NLI task. These vector representations, as well as the element-wise product and difference between them, are concatenated into a single vector m, which is then passed through a *Multi-Layer Perceptron* (MLP). To associate a probability to each possible class in the NLI task, a *softmax* function is applied on the output of the MLP.

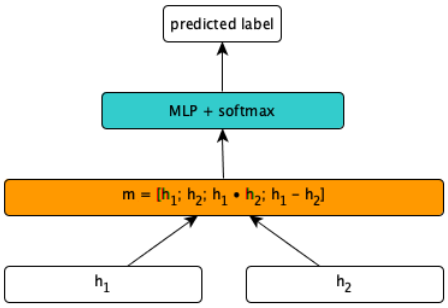

Figure 2.2: Classifier architecture in sentence vector-based models

Since the classifier in all sentence vector-based models is virtually the same (with only hyper-parameters such as layers size varying), the main difference between these models lies in the way they learn representations for the premises and hypotheses in RTE problems. Approaches to sentence encoding can be subdivided into three principal categories: *sequential*, *tree-based* and *self attention-based*.

Sequential sentence encoders are models that use *Recurrent Neural Networks* (RNNs) to learn representations for sentences. Examples of such approaches can be found in Conneau, Kiela, et al. (2017), where the authors investigate several architectures involving RNNs for sentence encoding. In particular, they propose in a first approach to encode sentences by passing the word embeddings that compose them through a unidirectional, single-layer GRU or LSTM network and taking the final state of the RNN as representation. In a second proposition, a *bidirectional LSTM* (bi-LSTM) is used, and *max* or *average pooling* is applied over each dimension of the network's hidden states to extract a fixed-length vector for a sentence.
In subsequent works (Nie and Bansal, 2017; Talman et al., 2018), more complex architectures using *stacked bi-LSTMs* with shortcut connections and max pooling over the output are used to encode sentences. The general structure of these stacked encoders is illustrated in figure 2.3 (with only minor differences between the implementations proposed in the two cited papers).
Table 2.4 summarises the reported accuracies of sequential encoders on SNLI and MultiNLI's test sets.

Model	**SNLI**	**MultiNLI-m**	**MultiNLI-mm**
LSTM (Conneau, Kiela, et al., 2017)	80.7	-	-
GRU (Conneau, Kiela, et al., 2017)	81.8	-	-
Bi-LSTM avg. (Conneau, Kiela, et al., 2017)	78.2	-	-
Bi-LSTM max. (Conneau, Kiela, et al., 2017)	84.5	-	-
Shortcut-stacked encoder (Nie and Bansal, 2017)	86.1	**74.6**	**73.6**
HBMP (Talman et al., 2018)	**86.6**	73.7	73.0

Table 2.4: Reported accuracy (%) of sequential sentence encoders on SNLI and MultiNLI's matched (MultiNLI-m) and mismatched (MultiNLI-mm) test sets

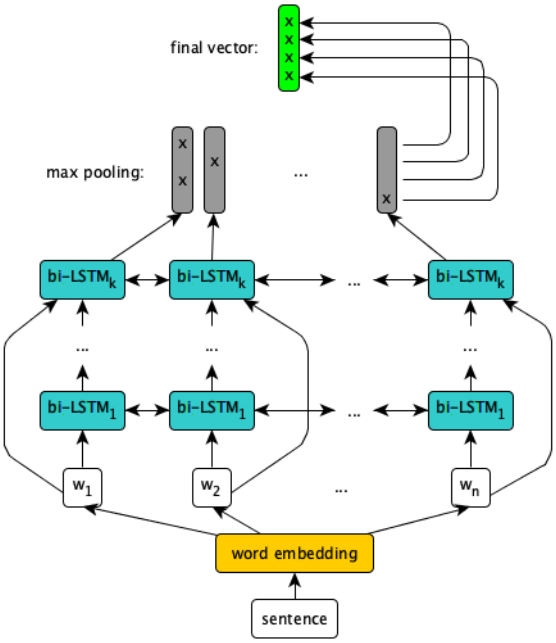

Figure 2.3: k-layers stacked bi-LSTMs with shortcut connections for sentence encoding

In *tree-based encoders*, the parse structure of sentences is used to learn representations for them. This way, syntactic information is directly taken into account by encoders while modelling the premises and hypotheses of NLI tasks.
In the *Tree-based Convolutional Neural Network* (TBCNN) (Mou et al., 2015), a *Convolutional Neural Network* (CNN) is applied over the dependency parse tree of a sentence to learn a representation for it. More specifically, Mou et al. propose in their paper to slide a set of *two-layers sub-tree feature detectors* over the parse tree of a sentence to learn *feature maps* for each word in it. The feature detectors are convolution filters specialised to capture information about specific grammatical relations between words and applied on every sub-tree in the dependency parse tree of a sentence. Once the detectors have been applied, the resulting feature maps are combined together by applying a max pooling operation over each of their dimensions and then passing the result through some *feed-forward* neural network. This produces a fixed-length vector that can be used in the final classification layer of the model.

Aside from CNNs, other types of networks can be applied on the parse structure of sentences to encode them. This is the case of *Tree-structured Recursive Neural Networks* (TreeRNNs), a special type of RNNs that work on binary parse trees and propagate information upstream along them, directly including syntactic information in the process of sentence encoding.
In a work by S. R. Bowman et al. (2016), the *Stack-augmented Parser-Interpreter Neural Network* (SPINN) is presented. Inspired by *shift-reduce parsing*, SPINN is a TreeRNN capable of both building the binary parse tree of a sentence and processing it in a single left-to-right pass over its tokens.
The same is true of the model proposed by Choi et al. (2017), which introduces a new kind of TreeRNN called *Gumbel Tree-LSTM* to learn the parse structure of a sentence as it is being read.

This capability of the two models to parse sentences as they learn their representations makes them particularly fast, because they only need one pass over a sentence to encode it, and they can work on batches of sentences instead of single sequences at a time (which is a big limitation of other TreeRNNs).
Table 2.5 summarises the reported classification accuracies of tree-based sentence encoders on SNLI's test set (no results are available on MultiNLI for those models).

Model	**Accuracy (%)**
TBCNN (Mou et al., 2015)	82.1
SPINN (S. R. Bowman et al., 2016)	83.2
Gumbel Tree-LSTM (Choi et al., 2017)	**86.0**

Table 2.5: Reported accuracy of tree-based sentence encoders on SNLI's test set

In *self attention-based encoders*, a special *attention* mechanism is used to build representations for sentences. As explained by Shen, Zhou, Long, Jiang, S. Pan, et al. (2017), attention is used to compute an alignment score between the elements of a sequence $x = [x_1, x_2, ..., x_n]$ and some *query* q. More specifically, an attention function $a(x_i, q)$ computes the degree of *similarity* or *dependency* between each $x_i \in x$ and q. A softmax function is then applied on the resulting scores to produce a probability distribution describing how likely each element x_i is to contribute to the information in the query.
Typically, the attention scores computed for a sequence x are used in some weighted operation involving the $x_i \in x$ (such as a sum) to build a summary of the relationship between x and q. In the specific case of sentence encoding, this idea is applied on the words of a sentence to learn a general representation for it (which can be seen as summarising the sentence's meaning). The name *self attention* (or *inner attention*) comes from the fact that attention is computed on the same sentence that is being encoded, as opposed to other situations in NLP where the mechanism is used on pairs of sentences to extract the dependencies between their elements.

There are multiple ways of using self attention to encode sentences' meanings in vector representations. Y. Liu et al. (2016) first pass the n word embeddings e_i of a sentence s through a bi-LSTM, which produces hidden states $h_i, i \in [1, .., n]$. They then apply average pooling on the h_i and use a feed-forward network to compute the attention between the resulting vector and the hidden states of the bi-LSTM. This produces weights α_i that are used in a weighted sum of the h_i to get a vector representation m for the sentence s. The architecture is illustrated in figure 2.4.
If we map Liu et al.'s attention mechanism to the one described in the previous paragraph, we see that the result of average pooling corresponds to the query vector q, the bi-LSTM's hidden states to the x_i, and the feed-forward network to the attention function $a(x_i, q)$.
Conneau, Kiela, et al. (2017) and Lin et al. (2017) inspire themselves from the attention mechanism described in the work by Liu et al., but they propose to use multiple attentions computed between the bi-LSTM's hidden states and learned query vectors, instead of a single attention with the result of average pooling as query. To force the attentions to focus on different parts of a sentence, a special penalisation term is used.

Shen, Zhou, Long, Jiang, S. Pan, et al. (2017) further develop the idea of self

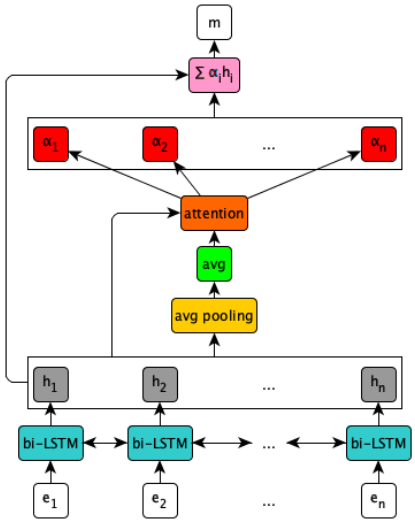

Figure 2.4: Inner attention mechanism (Y. Liu et al., 2016)

attention in their *Directional Self Attention Network* (DiSAN), which introduces the concept of *multi-dimensional self attention* (a form of attention where weights are computed for each dimension of a vector, instead of producing scalar attention scores). In a later publication, Shen, Zhou, Long, Jiang, Sen Wang, et al. (2018) present the *Reinforced Self Attention Network* (ReSAN), an improved DiSAN model that combines hard and soft attention mechanisms (Bahdanau et al., 2014; Xu et al., 2015) to learn representations for sentences.
In a paper by Chen, Zhu, Ling, Wei, et al. (2017b), a model using *gated attention* (a form of attention using the bi-LSTM's gates in its computation) is proposed. Chen, Ling, et al. (2018) present another approach making use of several forms of multi-dimensional self attention with *generalised pooling*. In other works, Munkhdalai and Yu (2017) propose a new neural architecture based on memory and self attention for sentence encoding, and Yoon et al. (2018) apply self attention on top of a CNN in their *Dynamic Self Attention* (DSA) network to learn representations. Finally, Im and Cho (2017) use the *Transformer* architecture (Vaswani et al., 2017) and a form of attention sensitive to the distance between words in their *Distance-based Self Attention Network*.

Table 2.6 summarises the reported accuracies of self attention-based models on the SNLI and MultiNLI test sets.

Model	**SNLI**	**MultiNLI-m**	**MultiNLI-mm**
Inner attention (Y. Liu et al., 2016)	83.3	-	-
Neural Semantic Encoder (Munkhdalai and Yu, 2017)	84.6	-	-
Structured Self Attentive Network (Lin et al., 2017)	84.4	-	-
Inner attention (Conneau, Kiela, et al., 2017)	82.5	-	-
Gated attention bi-LSTM (Chen, Zhu, Ling, Wei, et al., 2017b)	85.5	72.8	73.6
DiSAN (Shen, Zhou, Long, Jiang, S. Pan, et al., 2017)	85.6	71.0	71.4
Distance-based Self Attention Network (Im and Cho, 2017)	86.3	**74.1**	72.9
ReSAN (Shen, Zhou, Long, Jiang, Sen Wang, et al., 2018)	86.3	-	-
bi-LSTM generalised pooling (Chen, Ling, et al., 2018)	86.6	73.8	**74.0**
DSA (Yoon et al., 2018)	**87.4**	-	-

Table 2.6: Reported accuracy (%) of self attention-based sentence encoders on SNLI and MultiNLI's matched (MultiNLI-m) and mismatched (MultiNLI-mm) test sets

2.1.2.2 Sentence Matching Models

Natural language inference is a task that involves comparing pairs of sentences to predict the relation linking them. This means that interactions between premises and hypotheses play an essential role in the decision whether they entail each other or not. However, because the goal of sentence vector-based models is to learn representations for single sentences that can be used in downstream tasks, they do not capture any information about interactions between sentence pairs to recognise entailment.
Sentence matching models, on the contrary, do exactly that. While this makes them less applicable in transfer learning, it gives them a clear advantage on recognising textual entailment, as is reflected by their classification accuracy on NLI tasks.

There are multiple ways of modelling interactions between sentences, but almost all of them involve attention mechanisms similar to the one described in the previous sub-section.

In the paper *"Reasoning about Entailment with Neural Attention"* (Rocktäschel et al., 2016), the authors use two LSTMs with word-by-word attention to learn representations for the interactions between premises and hypotheses in NLI.
The architecture of the model is illustrated in figure 2.5. In the image, the $e_i^p, i \in [1, ..., l]$ and $e_j^h, j \in [1, ..., m]$ are word embeddings for the premise and hypothesis, respectively, $LSTM^p$ and $LSTM^h$ two LSTMs to encode them, and h_i^p and h_j^h the hidden states of $LSTM^p$ and $LSTM^h$. The model uses a feed-forward network to compute attention between each h_j^h (the encoded words of the hypothesis) and all the h_i^p (the encoded words of the premise) to produce attention weights α_{ij}. For each h_j^h, an attention vector r_j is then obtained by computing a weighted sum a_j of the attended h_i^p with the α_{ij} ($a_j = \sum_{i=1}^{l} \alpha_{ij} h_i^p$) and merging it with r_{j-1}, the attention vector computed for the previous word in the hypothesis ($r_j = a_j + tanh(W^r r_{j-1})$, W^r is a learnable parameter). Finally, the last attention vector r_m is taken as a representation of all the interactions between the premise and hypothesis and merged with the last hidden state h_m^h of $LSTM^h$. The result is passed through a classification layer with softmax activation to predict the label associated to the pair.

Subsequent works inspire themselves from the proposition by Rocktäschel et al. to build their own LSTMs with word-by-word attention for NLI.
Shuohang Wang and Jiang (2016) reuse the same architecture, but they pass each h_j^h and a_j through an additional layer they call *match-LSTM* (mLSTM) to merge their representations. They then use the last hidden state of the mLSTM for classification.
Sha et al. (2016) first encode the premise with a regular LSTM network and concatenate its hidden states in a matrix P. Then, they pass P and the word embeddings of the hypothesis through a special LSTM called *re-read LSTM* (rLSTM), which combines the encoded words of the hypothesis with a weighted sum of the vectors in P. The weights of the sum are computed with attention. They apply average pooling on the outputs of the rLSTM and use the result for classification.
P. Liu et al. (2016) propose something slightly different with their *coupled-LSTMs* (c-LSTMs), two inter-dependent LSTMs that encode the premise and hypothesis

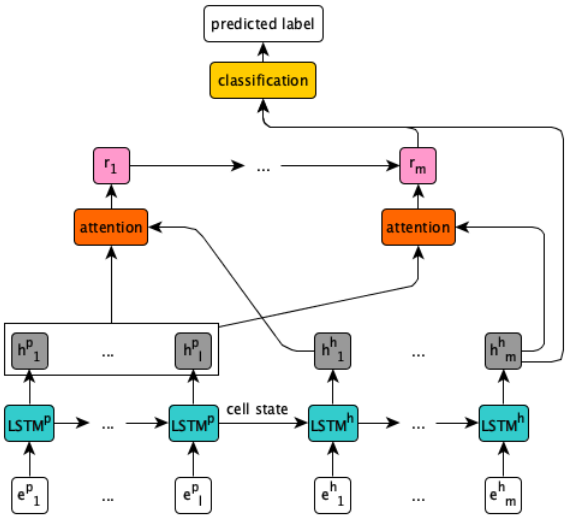

Figure 2.5: Word-by-word attention mechanism (Rocktäschel et al., 2016)

in a pair by using both their own previous hidden states and the ones from the other LSTM at different time steps to produce outputs. The authors stack multiple coupled-LSTM on top of each other to get their best performance on NLI with this approach.

The *attend-compare-aggregate* model proposed by Parikh et al. (2016) applies neural attention directly on the word embeddings of a sentence pair to perform classification. First, it uses a feed-forward network to compute attention scores between each word embedding in the premise, denoted $e^p_i, i \in [1, .., l]$, and those in the hypothesis, denoted $e^h_j, j \in [1, .., m]$. A softmax function is applied on the result, which produces attention weights α_{ij} that are used to compute, for each word e^p_i in the premise, a weighted sum of the words in the hypothesis $a_i = \sum_{j=1}^{m} \alpha_{ij} e^h_j$, and the inverse for each word e^h_j in the hypothesis, $b_j = \sum_{i=1}^{l} \alpha_{ij} e^p_i$. Then, each pair (e^p_i, a_i) and (e^h_j, b_j) is concatenated and passed through a feed-forward network G to produce comparison vectors $v^p_i = G([e^p_i; a_i])$ and $v^h_j = G([e^h_j; b_j])$. Finally, the v^p_i and v^h_j are aggregated by summing them, $v^p = \sum_{i=1}^{l} v^p_i$, $v^h = \sum_{j=1}^{m} v^h_j$, and the results are concatenated and passed through a final feed-forward network for classification, $\hat{y} = F([v^p; v^h])$. The complete architecture of the model is illustrated in figure 2.6.

After the publication by Parikh et al. in 2016, the idea of computing attention both from the premise to the hypothesis and from the hypothesis to the premise was reused in numerous other sentence matching models.
Chen, Zhu, Ling, Wei, et al. (2017a) introduce the *Enhanced Sequential Inference Model* (ESIM), a neural network that uses bi-LSTMs to encode premises and hypotheses and *bidirectional* attention to model interactions between them. At the time of its publication and for a while after it, ESIM remained state-of-the-art on the SNLI corpus. It was also the best performing model on MultiNLI when the data set was initially released. These are some of the reasons why it was chosen as a baseline in this work. More details on the model's architecture and performance are given in chapter 3.

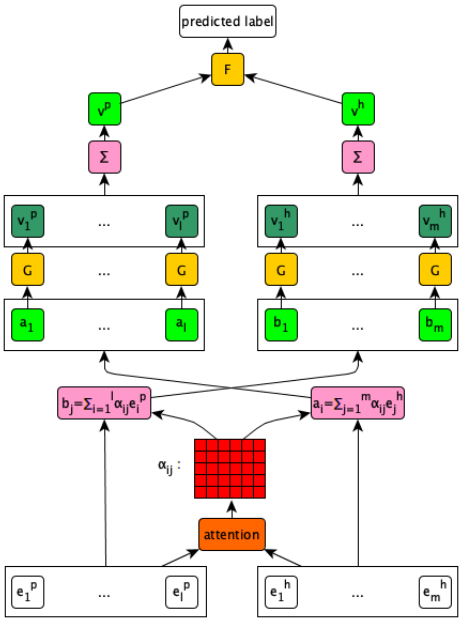

Figure 2.6: Attend-compare-aggregate architecture (Parikh et al., 2016)

Similar to ESIM, the *Bilateral Multi-Perspective Matching* (BiMPM) network (Z. Wang et al., 2017) is composed of a first bi-LSTM to encode premises and hypotheses in NLI pairs, a bidirectional attention layer, a second bi-LSTM to compose attention information and a final classification layer. Differences between the BiMPM and ESIM mostly lie in the way attention is computed and composed, but overall the architectures of the two are very close.
In another work, Tay et al. (2018) introduce the *Comprop Alignment-Factorised Encoders* (CAFE). The model also has an architecture similar to ESIM, but it composes attention information differently, with an *alignment factorisation layer*.
The *Dependant Reading Bidirectional LSTM* (DR-BiLSTM) model from Ghaeini et al. (2018) is yet another neural network which works in the same way as ESIM, with the exception that it uses special inter-dependant bi-LSTMs for the encoding of the premise and hypothesis instead of regular ones.
Finally, the *Densely-connected Recurrent and Co-attentive neural Network* (DRCN) (Kim et al., 2018) adopts an architecture comparable to ESIM's, but it stacks multiple RNNs and attention layers on top of each other. Auto-ecoders are used between stacked layers to reduce the representations dimensionality (since it grows with the number of layers). Figure 2.7, taken from the original paper, illustrates the concepts of the model. The top-right part of the image shows the specific number and disposition of layers used by the authors in the publication.

In the *Densely Interactive Inference Network* (DIIN) (Gong et al., 2018), the authors use a *highway network* (Srivastava et al., 2015) followed by a self attention mechanism to encode the premise and hypothesis of a sentence pair. A form of multi-dimensional attention is computed between the representations obtained with highway networks, producing a 3-dimensional attention matrix. A convolutional feature extractor is then applied on it to retrieve information about interactions between the sentences, and the result is flattened and passed through a MLP with softmax to predict the class associated to the pair of inputs.

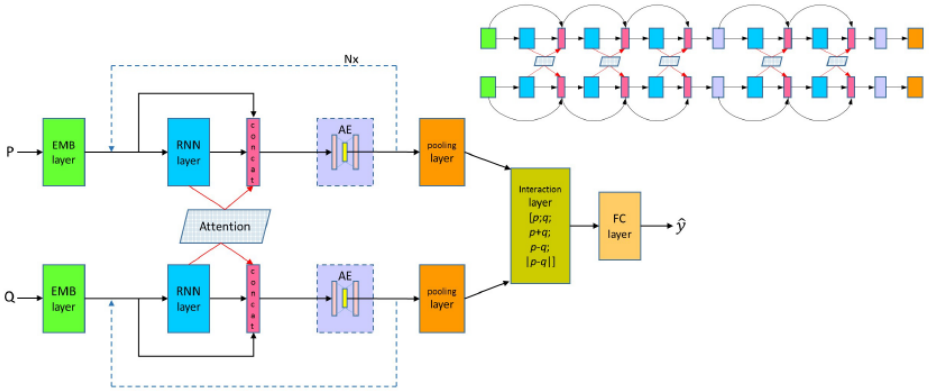

Figure 2.7: DRCN architecture (Kim et al., 2018)

Only very few papers investigate the use of external knowledge for NLI. All of them propose to use sentence matching models with the inclusion of some outside information to improve their performance on recognising entailment.
Chen, Zhu, Ling, Inkpen, et al. (2018) introduce the *Knowledge-based Inference Model* (KIM). The model follows the general architecture of ESIM, but information about the lexical relations between words in the premise and hypothesis is additionally used to improve the model's performance. In particular, lexical relations between words such as synonymy, hypernymy or antonymy are extracted from *Wordnet* (Miller, 1995) and used in the attention layer.
In another paper (Zhang et al., 2018), the authors propose to use *Semantic Role Labelling* (SRL), a task where the objective is to predict the predicate-argument relations in a sentence, to improve an existing model on NLI. The authors apply SRL on premises and hypotheses to learn the semantic roles for the words they contain, and the results are used as additional information in the ESIM model, which improves its performance.

Table 2.7 summarises the reported accuracies of sentence matching models on SNLI and MultiNLI's test sets.

Model	**SNLI**	**MultiNLI-m**	**MultiNLI-mm**
Word-by-word attention (Rocktäschel et al., 2016)	83.5	-	-
Match-LSTM (Shuohang Wang and Jiang, 2016)	86.1	-	-
rLSTM (Sha et al., 2016)	87.5	-	-
Stacked TC-LSTMs (P. Liu et al., 2016))	85.1	-	-
Attend-Compare-Aggregate (Parikh et al., 2016)	86.3	-	-
ESIM (Chen, Zhu, Ling, Wei, et al., 2017a)	88.0	76.8	75.8
ESIM + Tree-LSTM (Chen, Zhu, Ling, Wei, et al., 2017a)	88.6	-	-
BiMPM (Z. Wang et al., 2017)	87.5	-	-
CAFE (Tay et al., 2018)	88.5	78.7	77.9
DR-BiLSTM (Ghaeini et al., 2018)	88.9	-	-
DRCN (Kim et al., 2018)	88.9	**80.6**	**79.5**
DIIN (Gong et al., 2018)	88.0	80.0	78.7
KIM (Chen, Zhu, Ling, Inkpen, et al., 2018)	88.6	77.2	76.4
SRL (Zhang et al., 2018)	**89.1**	-	-

Table 2.7: Reported accuracy (%) of sentence matching models on SNLI and MultiNLI's matched (MultiNLI-m) and mismatched (MultiNLI-mm) test sets

2.1.2.3 Transfer Learning Approaches

In *transfer learning*, models that were first trained on some task are reused and fine-tuned to perform other tasks. The approach is often used in situations where resources are very scarce for a given problem, but there are large amounts of training data available for some other, related task. In those cases, a model is first trained on the objective where lots of data are available, and its parameters are then fine-tuned for the problem with fewer resources.

Over the past years, there have been numerous examples of successful applications of transfer learning to deep learning models. While earlier examples have mostly been in computer vision (thanks to the release of the huge *ImageNet* data set (Russakovsky et al., 2015)), more recent applications to NLP tasks have also shown to provide impressive performance gains.

In NLP, almost unlimited resources are available in the form of unlabelled, free text. This data can be used in the unsupervised training of deep learning models to capture information about the general form and structure of language. Models trained in this manner can then be later fine-tuned for more specific tasks requiring human annotated resources, which are much more difficult to produce and hence much scarcer.

In the specific case of NLI, although reasonably large quantities of data are available for training, the application of transfer learning has allowed models to reach significantly higher classification accuracy than the previous state-of-the-art on famous data sets like SNLI or MultiNLI. These results show that pre-training models on unsupervised tasks to allow them to better model language seems to also make them more efficient on natural language understanding.

There also exist cases where transfer learning is used between natural language inference and other supervised tasks. In those situations, performance improvements show that information about linguistic phenomena captured in other tasks and overlooked by NLI can help models recognise entailment with more accuracy.

Transfer Learning from Supervised Tasks

Approaches where transfer learning is applied between some supervised task and NLI include the models proposed by McCann et al. (2017) and B. Pan et al. (2018).

McCann et al. (2017) propose to apply transfer learning between *Machine Translation* (MT) and other NLP problems by learning special *Context Vectors* (CoVe) that can be reused in downstream tasks. The general idea of the model, illustrated in figure 2.8 (taken directly from McCann et al.'s paper), is to first train the sentence encoder of a MT task to learn context sensitive representations for words so they can be translated accurately, and then reuse it in other downstream tasks to encode input words.
The insight behind this approach is that, in many NLP problems, models need representations for words that are specific to the contexts in which they appear to perform well. "Traditional" word embedding methods, however, only produce single vectors for words that are context-independent.

In machine translation, it is only after word embeddings are passed through a model's encoder that their representations become context specific. Because MT is a problem where large quantities of parallel data are available for training, it also happens to be well adapted for transfer learning. This justified the use of a pre-trained MT encoder to contextualise words in dowstream tasks.
For the particular problem of natural language inference, McCann et al. show in their paper that using CoVe improves the accuracy of the model they train on the SNLI corpus.

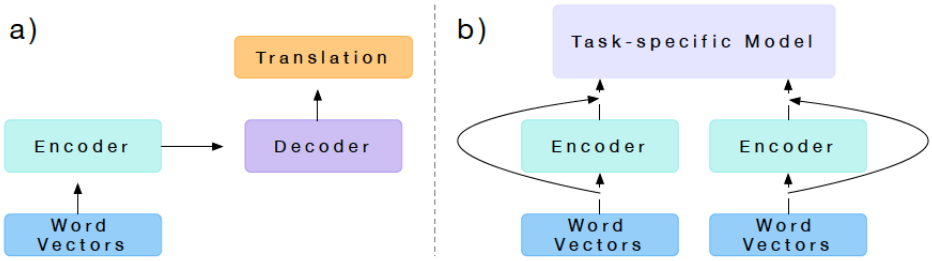

Figure 2.8: Transfer learning in CoVe (McCann et al., 2017)

In the *Discourse Marker Augmented Network* (DMAN) (B. Pan et al., 2018), the authors propose to apply transfer learning between the supervised task of *Discourse Marker Prediction* (DMP) and NLI.
In DMP, the objective is to predict the *discourse marker* which connects the two halves S_1 and S_2 of a sentence S. Discourse markers are words that carry information about the relation between parts of a sentence, such as "and", "or" or "but". The authors of DMAN underline in their paper how these words *"intuitively correspond to the intent of NLI, such as 'but' to contradiction, 'so' to entailment, etc."* (B. Pan et al., 2018), which is why they choose to transfer knowledge from DMP to RTE. To do so, they first train a sentence encoder on a DMP task, and then integrate it in a model specifically designed to recognise entailment. The sentence encoder's parameters are fine-tuned during training on the NLI task. With this approach, new state-of-the-art results were obtained at the time of publication.

Transfer Learning from Unsupervised Tasks

Language modelling is a common task in NLP where systems called *Language Models* (LMs) are trained to learn about the structure of language in an unsupervised manner. Usually, the objective for LMs is to predict the probability of the next word in a sentence given the previous ones. Formally, if $s = [w_1, w_2, ..., w_n]$ is a sentence of n words w_i, the goal of a language model is to predict $P(w_i|w_1, ..., w_{i-1}), \forall i \in [1, ..., n]$. Modern LMs learn to predict such probabilities statistically on large corpora of unlabelled data, and the current state-of-the-art is obtained with deep neural networks. It is generally accepted that, with sufficient amounts of data available for training, LMs are able to learn good representations for language. Since unsupervised text exists in almost unlimited quantities, this makes language modelling particularly appropriate for transfer learning.

In the paper "Deep Contextualised Word Representations", Peters et al. (2018) propose to use transfer learning to contextualise word vectors in NLP tasks. Their

approach to transfer knowledge from one task to the other is similar to McCann et al.'s with CoVe: an encoder is first trained on some particular task to contextualise word embeddings, and it is then integrated in other models to encode their input words. However, the task used by Peters et al. for pre-training is very different from the one in CoVe. In their *Embeddings from Language Models* (ELMo), Peters et al. first train a deep *bidirectional Language Model* (biLM) on some unsupervised language modelling task and then use a linear combination of the biLM's internal states to represent input words in downstream tasks.
ELMo's authors show that using a combination of the biLM's internal states produces richer word representations than simply taking the network's output, for example, because different layers of a biLM capture different levels of information about language (lower layers are often more focused on structure and syntax, while layers at the top usually learn about meaning).
When they integrate ELMo in existing models for various NLP tasks, and in particular the well-known ESIM for NLI, Peters et al. report increases in classification accuracy, justifying their approach.

In the paper "Improving Language Understanding by Generative Pre-Training", Radford et al. (2018) present the *Generative Pre-training Transformer* (GPT), a language model based on the *transformer* architecture (Vaswani et al., 2017).
In order to apply transfer learning, the GPT is first trained on a language modelling objective with unlabelled data, and it is later fine-tuned for various natural language understanding tasks. Figure 2.9 (taken from the GPT paper) illustrates a high-level view of the model's architecture on the left, as well as a representation of the way the inputs for different tasks are modified to fine-tune GPT on the right. In the particular case of entailment, the premise and hypothesis of a NLI task are concatenated into a single sequence (separated by special delimiters), the result is passed through the transformer language model, and the model's final output is used as input in a linear classification layer. With this approach, Radford et al. manage to obtain new state-of-the-art result at the time of publication.

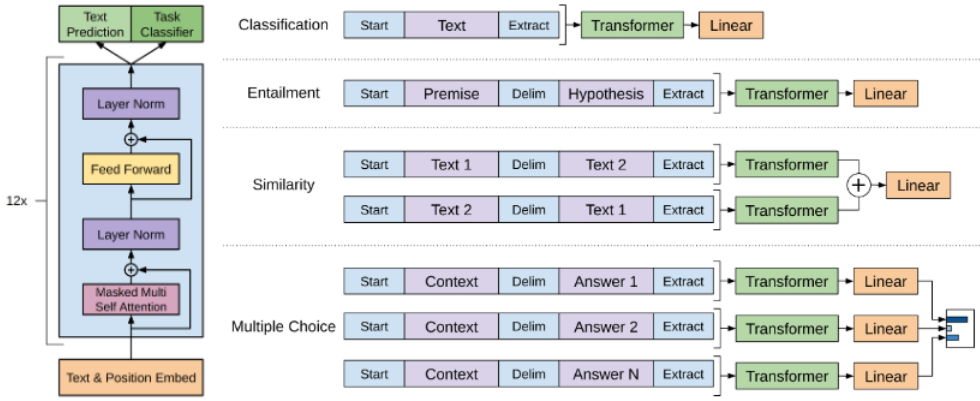

Figure 2.9: Architecture of the GPT and transfer to other tasks (Radford et al., 2018)

Devlin et al. (2018) make the observation that because Radford et al.'s GPT uses a regular language modelling objective during pre-training, the representations it learns are not bidirectional, which limits their representational power. Indeed, because the "traditional" objective of a language model is to predict the next word

in a sentence based on the ones it's already seen, truly bidirectional models cannot be used for this task, as they have already seen the next word in a sentence at any time step, which makes predictions trivial and prevents them from learning anything.

While bidirectionality can be mimicked with the combination of two separate left-to-right and right-to-left LMs (as is done in ELMo), Devlin et al. argue that this approach is sub-optimal compared to a truly bidirectional model. For this reason, they introduce the *Bidirectional Encoder Representations from Transformers* (BERT), a fully bidirectional language model for transfer learning in NLU.
In order to make bidirectionality possible in BERT, Devlin et al. devise a new language modelling objective: the *Masked Language Model* (MLM). In the MLM, random words in a language model's input are masked, and the goal is to predict them. This method makes bidirectionality possible, because models don't know the words that are masked in advance, even if they have already seen them in the input. In addition to the MLM, Devlin et al. also introduce a next sentence prediction task during the pre-training of their model. In this task, pairs of sentence are extracted from the language model's training data, with part of them following each other in the text, and others selected at random. The goal for the model is then to predict if the sentences it receives as input follow each other in the text or not.

For NLI, transfer learning with BERT is done as illustrated in figure 2.10 (taken directly from the original paper). The premise and hypothesis in a sentence pair are concatenated (with a separator token between them), and a special *class* ($[CLS]$) token is appended at the beginning of the sequence. For classification, an additional output layer is integrated at the end of the model to predict the relation between the sentences passed as input.

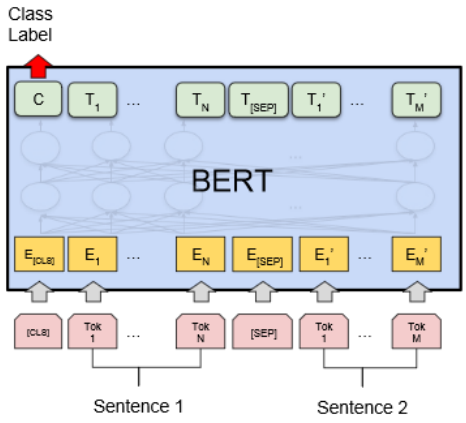

Figure 2.10: Transfer learning to NLI with BERT (Devlin et al., 2018)

In order to make comparison with the work by Radford et al. possible, Devlin et al. propose a version of BERT with approximately the same number of parameters as the GPT, named $BERT_{BASE}$. In addition, another larger version of the model called $BERT_{LARGE}$ is trained and tested. Both significantly outperform all other existing models, and $BERT_{LARGE}$ provides impressive new state-of-the-art results which show the power of transfer learning from bidirectional language models to NLU. Table 2.8 summarises the reported accuracies of transfer learning approaches on SNLI and MultiNLI.

Model	SNLI	MultiNLI-m	MultiNLI-mm
CoVe (McCann et al., 2017)	88.1	-	-
DMAN (B. Pan et al., 2018)	88.8	78.9	78.2
ELMo (Peters et al., 2018)	88.7	-	-
GPT (Radford et al., 2018)	**89.9**	82.1	81.4
$BERT_{BASE}$ (Devlin et al., 2018)	-	84.6	83.4
$BERT_{LARGE}$ (Devlin et al., 2018)	-	**86.7**	**85.9**

Table 2.8: Reported accuracy (%) of transfer learning approaches on SNLI and MultiNLI's matched (MultiNLI-m) and mismatched (MultiNLI-mm) test sets

2.2 Lexical Entailment

In this section, we describe the notion of *lexical entailment* and explore related work on the subject. Since there exists a considerable literature on this topic, we restrict ourselves here to modern state-of-the-art approaches that propose lexical entailment metrics or special word embeddings that can easily be incorporated into the baseline model we chose for this work.

2.2.1 Definition

Lexical Entailment (LE) is a relation similar to that of textual entailment, but at the word level rather than between sentences. A word is said to be entailed by another when its meaning can be reasonably inferred from it. In practice, this means that a word is entailed by another when it can replace it in a sentence without losing the sentence's meaning or making it more specific.

As is pointed out by Geffet and Dagan (2005), LE actually *"corresponds to several lexical semantic relations, such as synonymy, hyponymy and some cases of meronymy"*. For example, when the word *people* is used instead of its *meronym*[7] *faces* in the sentence *there were familiar faces at the reunion*, it can be said that *people* is entailed by *faces*. This affirmation might however not hold in other situations where the words' meanings are different: we can never say that *people* is entailed by *faces* when we are talking about the verb or the *face of the moon*, for example. This demonstrates how LE is a relation dependent on context.

In some publications (Vulić, Gerz, et al., 2017), the term lexical entailment is used to describe the more restrictive relation of *hypernymy*. Hyponymy/hypernymy corresponds to the *is-a* relationship between *hyponyms* and *hypernyms* and can be seen as a hierarchical categorisation of words' meanings. For example, the word *dog* is a hyponym of *animal*, because all dogs belong to the animal category, and a hypernym of *dachshund*, because it refers to a specific breed of dogs.

Contrarily to our definition of LE, hyponymy is not tied to a context and is defined for all the different meanings of a word. For example, the word *mouse* is considered to be a hyponym of *animal* no matter what context it appears in, because

[7] *Meronymy* corresponds to the *part-of* lexical relationship. For example, the word *tail* is said to be a *meronym* of dog, because it denotes a part or constituent of it.

one of its possible meanings is a small rodent (and even though in some other situation it could mean a part of a computer). Hence, with our definition, we can say that lexical entailment includes but is not limited to the stricter relation of hyponymy. In a sense, it can be said that hyponymy/hypernymy entails lexical entailment, and that it is therefore sufficient but not necessary to determine entailment (a hypernym is always entailed by its hyponyms).

The elements presented above illustrate how tightly related lexical and textual entailment are, and why we expect that the inclusion of measures of hyponymy and LE in natural language inference systems should benefit them.

2.2.2 Tasks and Data Sets

In order to evaluate the ability of models to recognise lexical entailment, a number data sets have been proposed over the years.

In the paper titled *"How we BLESSed distributional semantic evaluation"*, Baroni and Lenci (2011) introduce the BLESS data set, a collection of 26,554 word pairs connected through various lexical relations, such as *co-hyponymy*, *hypernymy* or *meronymy*.
To build BLESS, Baroni and Lenci selected 200 concrete English nouns in the singular form describing both living and non-living entities and covering a wide range of themes (objects, animals, vehicles, ...). They then paired those with other words (nouns, verbs and adjectives) connected to them through various lexical relations, producing triplets like `dog-HYPER-animal`, for example. In addition to related words, random unrelated pairs were also introduced in BLESS as "negative" examples (examples of pairs that should be recognised as non-related).
The resulting data set provides a way to evaluate the ability of distributional semantic models to both recognise whether two words are related and, if they are, to determine what relation connects them.

In the paper *"Learning to distinguish hypernyms and co-hyponyms"*, Weeds et al. (2014) use the BLESS data set to evaluate the model they present. Since they are only interested in hypernymy and co-hyponymy, they restrict their evaluation to the word pairs connected through these particular relations in BLESS. However, in order to also enable the evaluation of models on their ability to distinguish hypernymy from other lexical relationships, they also include an equal number of word pairs either unrelated or connected through different relations in their data. As a result, a new data set, sometimes referred to as WBLESS in later works (Kiela et al., 2015), is produced.

Another data set inspired by BLESS, coined BiBLESS, is proposed by Kiela et al. (2015). The corpus consists in word pairs from BLESS and WBLESS that are either connected through hypernymy (labelled with -1), hyponymy (labelled with 1) or some other relation (labelled with 0). Models can in this case not only be evaluated on their ability to distinguish hyponyms/hypernyms from others pairs (a task known as *hypernymy detection*), but also on their capability to recognise the direction of hyponymy/hypernymy (known as *hypernym directionality*).

Finally, Vulić, Gerz, et al. (2017) introduce the *HyperLex* data set, a collection

of 2,616 word pairs (2,163 noun pairs and 453 verb pairs) graded on a scale from 0 to 10 for lexical entailment (defined here in its more restricted sense of hypernymy). The grade associated to each pair in the set describes the strength of the hypernymy relationship between the words that compose it, with 0 denoting no entailment at all and 10 representing the strongest entailment relationship possible.
To build HyperLex, its authors selected word pairs covering a wide range of themes from the *USF norms* data set (Nelson et al., 2004) and annotated them with lexical relations automatically extracted from WordNet (Miller, 1995). They then presented those pairs to human annotators on a crowdsourcing platform and asked them to grade them on a scale from 0 to 6 for the relation of lexical entailment. After validating the results, the grades associated to the pairs where re-scaled linearly from 0 to 10.
Vulić et al. justify their choice to use *graded lexical entailment* in HyperLex by citing theories from cognitive science. They argue that humans reason about entailment in a gradual rather than binary way, which implies that graded lexical entailment data is necessary to model the relationship accurately. To back their proposition, they analyse inter-annotator agreement in their data set and perform a number of comparisons between the grades obtained in HyperLex and the relations they extracted from WordNet for the pairs that compose it. With this approach, they show that the scores in their data set are highly reflective of human perception of lexical entailment, proving their hypotheses right and demonstrating the quality of their corpus for the evaluation of LE models.

2.2.3 Lexical Entailment Models

In order to model and measure lexical entailment between words, a number of metrics, specialised vector spaces and word embeddings have been proposed over the years. These approaches can be regrouped in two main categories: unsupervised and supervised. The following sub-sections explore the subset of models in these categories that were considered in this work to be integrated with LEAN.

2.2.3.1 Unsupervised Approaches

Unsupervised approaches to lexical entailment often propose metrics to compute entailment between existing word embeddings, or new vector spaces learned with special objectives on unlabelled data.

An advantage of unsupervised approaches is that they usually suffer less from over-fitting and offer better generalisation power than their supervised counterparts. However, their performance on lexical entailment benchmarks is often worse, and they *"are unfit for [the] task [of graded lexical entailment] and unable to surpass the performance of simple frequency baselines"* (Rei et al., 2018). This is why supervised methods were also considered in this work.

In the paper *"A Vector Space for Distributional Semantics for Entailment"*, Henderson and Popa (2016) present a vector space providing *"a formal foundation for a distributional semantics of entailment"* (Henderson and Popa, 2016).

The framework they propose is based on the concept of vectors of binary features modelling what information is or isn't known in a word (rather than what information is true or false, like in most other distributional semantic vector spaces). This choice is justified by the fact that entailment is an asymmetric relation determined by knowledge about specific features, rather than by observations of their truth.

In particular, a word x can be said to entail another word y if and only if everything known in y is also known in x. The inverse is not true, and something known in x but unknown in y still means that x can entail y. Therefore, if x and y are represented by vectors of n binary features indicating whether some information is known (1) or not (0), the probability that x entails y is given by:

$$P(x \Rightarrow y \mid x, y) = \prod_{k=1}^{n} (1 - (1 - x_k)y_k)$$

To learn a model of lexical entailment from data, Henderson and Popa propose to approximate these binary feature vectors by reasoning with distributions over them. However, they observe that the dimensions of the vectors are not independent. For example, a given dimension could be modelling knowledge about a feature being true, while another could do so for the same feature being false. In that case, the two dimensions would be in a situation of mutual exclusion and therefore not independent. Because of this kind of inter-dependencies between features, the prior probability distributions over the vectors are not factorised, which makes exact inference on them intractable.

To get around this issue, Henderson and Popa propose a mean-field approximation where it is assumed that the posterior probabilities of the vectors are factorised. They underline how *"in practice, this is a much weaker assumption than assuming the prior is factorised"* (Henderson and Popa, 2016). Developing on this idea, they come up with several entailment operators for their proposed vector space and go on to re-interpret the *Word2Vec* model (Mikolov et al., 2013) *"as approximating an entailment-based model of the distributions of words in contexts"* (Henderson and Popa, 2016). They then proceed to evaluate the quality of their proposed space on the WBLESS data set and show that it performs better than previous work on both hypernymy detection and direction classification.

In a later publication, Henderson (2017) reuses the ideas from the work by Henderson and Popa (2016) to learn new word embeddings specialised for lexical entailment. To train the embeddings, the Word2Vec model is used, but its objective is replaced with a function derived from the entailment operator defined in (Henderson and Popa, 2016):

$$log(P(x \Rightarrow y)) \approx X \circledcirc Y = \sigma(-X) \cdot log(\sigma(-Y))$$

where X, Y are the vector representations for the words x and y, and σ is the *sigmoid* function. The newly learned word embeddings are called *Word2Hyp*, and entailment between them can be computed by using the operator described above.

Again, to assess the performance of Word2Hyp, Henderson evaluates it on the WBLESS data set. Results on both the hypernymy detection and direction classifi-

cation tasks show that the model reaches new state-of-the-art results at the time of its publication.

2.2.3.2 Supervised Approaches

Most supervised approaches to lexical entailment propose mappings from existing vector spaces to new ones learned with the help of manually labelled LE data. As mentioned in the previous sub-section, these methods usually perform better than unsupervised ones on LE benchmarks, in particular for graded lexical entailment tasks.

In the paper *"Specialising Word Vectors for Lexical Entailment"* (Vulić and Mrkšić, 2018), the authors present the *Lexical Entailment Attract-Repel* (LEAR), a method to transform existing vector spaces to make their topology more reflective of the LE relation. To do so, the *attract-repel* algorithm (Mrkšić et al., 2017) is used with a special cost function combining similarity and lexical entailment terms.

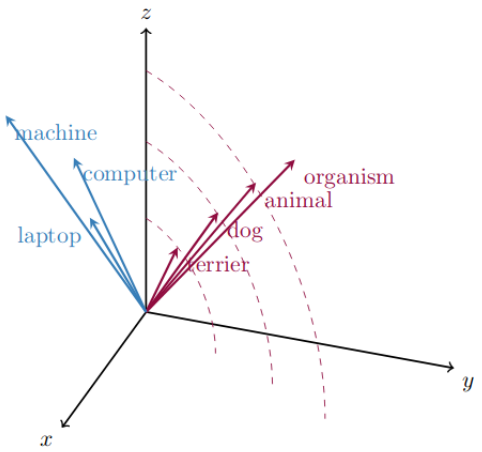

Figure 2.11: Effect of the LEAR algorithm on a transformed vector space (Vulić and Mrkšić, 2018)

The effect of LEAR is illustrated in figure 2.11, extracted from the paper by Vulić and Mrkšić (2018). Vectors for words that are semantically similar are brought closer together by reducing the *cosine distance* between them, while vectors for dissimilar words are moved away from each other. The lexical entailment term of the cost function also changes the norms of word vectors: more "general" words see their norm grow, while more "specific" ones see it shrink.

With this approach, it becomes possible to accurately predict the degree of lexical entailment between two words by using the function:

$$I_{LE}(x, y) = dcos(x, y) + D_j(x, y)$$

where *dcos* is the cosine distance and D_j a function measuring the difference between x and y's norms. A word represented by a vector y that is close to another vector x and has a larger norm is expected to be entailed by the word represented by x, with the degree of entailment between the words defined both by how close their vectors are and how much larger the norm of y is compared to that of x.

Vulić and Mrkšić evaluate LEAR's performance on the BLESS data set for LE direction classification, on WBLESS for LE detection, and on BiBLESS for both. They also test the model on Hyperlex for graded lexical entailment. On all four benchmarks, new state-of-the-art results are reached at the time of publication. However, although LEAR proves to be an efficient method to predict lexical entailment, an issue subsists with the algorithm: only the embeddings for words seen during training are transformed, with the rest of the vector space remaining unchanged.

The *Supervised Directional Similarity Network* (SDSN) (Rei et al., 2018) proposes to solve this problem. The model consists in a deep feed-forward neural network trained in a supervised manner to predict graded lexical entailment.

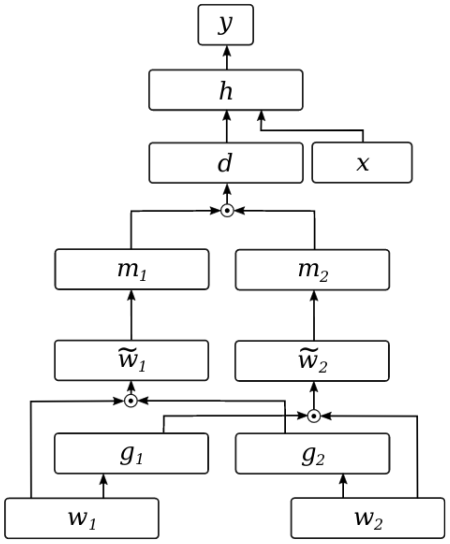

Figure 2.12: Architecture of the SDSN (Rei et al., 2018)

The architecture of the SDSN is illustrated in figure 2.12 (taken from the paper by Rei et al. (2018)). In the image, w_1 and w_2 are pre-trained, general purpose word embeddings that are passed through g_1 and g_2, two simple feed-forward layers with *sigmoid* activation functions that act as gating layers to learn representations for each word conditioned on the other. Component-wise multiplication is applied between each word embedding and the output of the gating layer for the other word, and the results are passed through mapping layers m_1 and m_2 (feed-forward layers with *tanh* activation). Then, the mapped vectors are multiplied component-wise, and the result is passed through a hidden feed-forward layer with *tanh* activation and an output layer. Finally, the scalar value output by the network is re-scaled between 0 and the desired maximum lexical entailment score S with a scaled sigmoid function.

To boost the performance of the SDSN, Rei et al. also propose to include sparse distributional features in the training procedure and to use additional supervision. With all of this, they report the best results to date on the Hyperlex data set.

An advantage of the SDSN over LEAR is that, instead of specialising individual word embeddings for lexical entailment, the model learns a mapping function from one vector space to another which can be applied not only on the words seen during training, but also on any other embedding of the input space.

Chapter 3

Baseline: The ESIM Model

The *Enhanced Sequential Inference Model* (ESIM) (Chen, Zhu, Ling, Wei, et al., 2017a) is a sentence matching model for NLI that reached state-of-the-art results at the time of its publication and has stayed among the top performing systems since then. The architecture of the model is simple yet powerful, and its integration with new components can usually be done in a seamless way. Thanks to this, the model has been reused, modified and augmented in many other publications (Chen, Zhu, Ling, Inkpen, et al., 2018; Peters et al., 2018; Williams et al., 2018; Zhang et al., 2018), often with the outcome of seeing its performance increased even further.

These elements led us to choose ESIM as a baseline for this work and to use its architecture as a basis for the new model and experiments we describe in chapter 4.

The rest of this chapter is structured as follows. In section 3.1, a detailed description of ESIM's architecture is given. In section 3.2, our implementation of the model with *PyTorch* is briefly presented. Finally, section 3.3 provides information about the specific parameters used to train the model, the experiments led on it and the results that were obtained with it on three famous benchmarks for NLI.

3.1 Description of the Model

A high level view of ESIM's architecture is presented in figure 3.1 below. As described in the original paper, the model can be subdivided into three main components that serve different purposes in the task of predicting inference between natural language sentences.

The first component, called *input encoding*, is used to specialise the embeddings for the words in the premises and hypotheses of some NLI task to make them *context dependent*.
Then, *local inference modelling* computes a form of *soft attention* between the specialised word embeddings to model interactions between them.
Finally, the *inference composition* component merges the representations learned by local inference modelling and uses the result to predict the classes associated to sentence pairs.

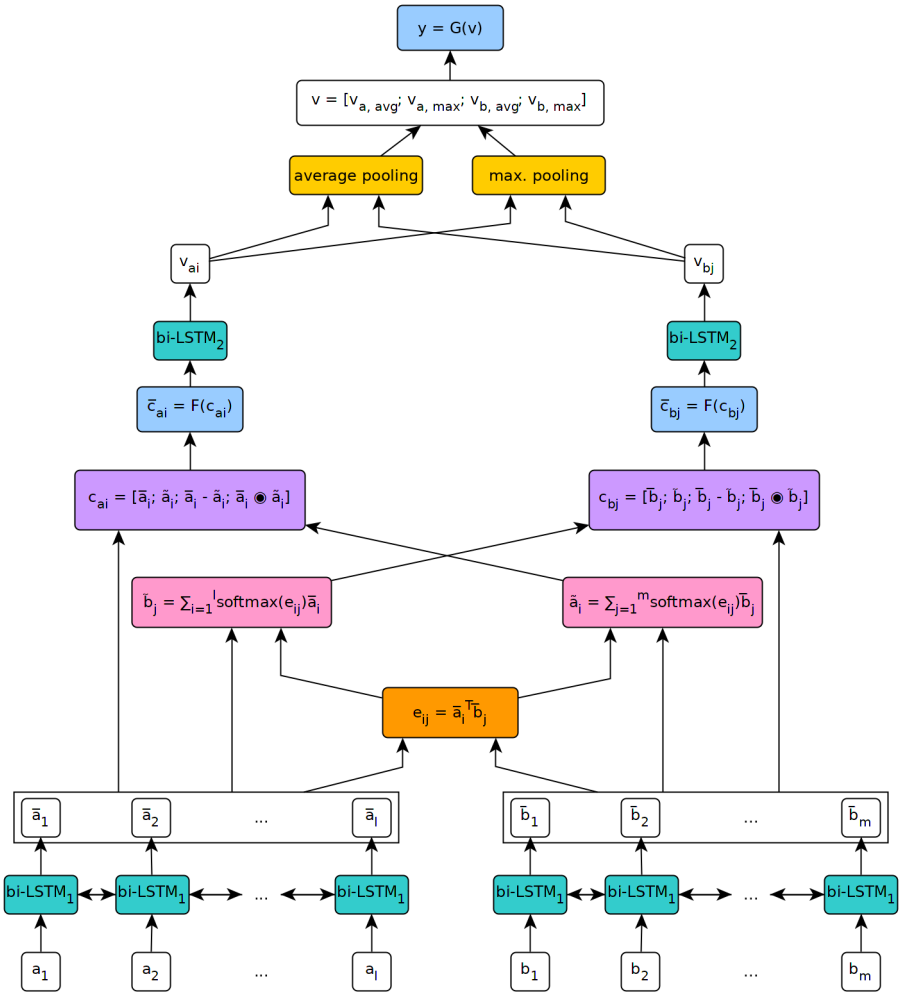

Figure 3.1: Architecture of the *Enhanced Sequential Inference Model* (ESIM) (Chen, Zhu, Ling, Wei, et al., 2017a)

3.1.1 Input Encoding

To encode the words that compose the premises and hypotheses in some NLI task, pre-trained word embeddings are used in ESIM. In figure 3.1, these word embeddings are represented by $a_i, i \in \{1, ..., l\}$ for the premise a of a sentence pair and $b_j, j \in \{1, ..., m\}$ for the hypothesis b.

"General purpose" pre-trained word embeddings such as *GloVe* (Pennington et al., 2014) or *Word2Vec* (Mikolov et al., 2013) usually learn representations for words that are independent from context. Hence, to make the a_i and b_j context dependent in ESIM, Chen, Zhu, Ling, Wei, et al. (2017a) pass them through a bi-LSTM and take the resulting hidden states as their new representations.
Bi-LSTMs consist in the concatenation of the hidden states of a *Long Short Term Memory* (LSTM) network applied both from left-to-right and from right-to-left on some input sequence (for a detailed description of the inner workings of LSTMs, we refer the reader to the work by Hochreiter and Schmidhuber (1997), or Christopher Olah's blog[1]).

[1] `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`

If we denote with $LSTM(x_1, x_2, ..., x_i)$ the output of a left-to-right LSTM at time step i for a sequence $x = \{x_1, x_2, ..., x_n\}$, and with $LSTM(x_n, x_{n-1}, ..., x_i)$ the output of the same LSTM applied from right-to-left on x, the contextualised representations of a_i and b_j in ESIM are obtained with:

$$\bar{a_i} = [LSTM_1(a_1, a_2, ..., a_i); LSTM_1(a_l, a_{l-1}, ..., a_i)]$$
$$\bar{b_j} = [LSTM_1(b_1, b_2, ..., b_j); LSTM_1(b_m, b_{m-1}, ..., b_j)]$$

where the symbol ";" inside square brackets indicates the concatenation of vectors.

3.1.2 Local Inference Modelling

Once the words in the premise and hypothesis passed as input to ESIM have been encoded, a form of soft attention is computed between them to model their interactions. The objective of this step is to try and detect inferences between the words in the two sentences which could help the system determine the class associated to the pair.

First, the similarity between the encoded embeddings of the premise $\bar{a_i}, i \in \{1, ..., l\}$ and those of the hypothesis $\bar{b_j}, j \in \{1, ..., m\}$ is computed with the dot product (corresponding to an unnormalised measure of their cosine similarity):

$$e_{ij} = \bar{a_i}^T \bar{b_j}$$

Then, the attention weights e_{ij} are used to compute a representation for each word in the premise conditioned on those in the hypothesis, and vice-versa:

$$\tilde{a_i} = \sum_{j=1}^{m} \frac{exp(e_{ij})}{\sum_{k=1}^{m} exp(e_{ik})} \bar{b_j}, \forall i \in \{1, ..., l\}$$
$$\tilde{b_j} = \sum_{i=1}^{l} \frac{exp(e_{ij})}{\sum_{k=1}^{l} exp(e_{kj})} \bar{a_i}, \forall j \in \{1, ..., m\}$$

In the formulas above, the attention weights computed between each word in the premise and all of those in the hypothesis are transformed to a probability distribution with a softmax function, and they are then used in a weighted sum of the encoded words of the hypothesis. The same is done the other way around for every word in the hypothesis.
This way, information relevant to each word in the premise and hypothesis is selected in the other sentence to learn a representation of the interactions between the two.

To model inference between sentence pairs, the encoded and conditioned representations of words are concatenated in new vectors, along with their difference and component-wise product:

$$c_{a_i} = [\bar{a_i}; \tilde{a_i}; \bar{a_i} - \tilde{a_i}; \bar{a_i} \odot \tilde{a_i}], \forall i \in \{1, .., l\}$$
$$c_{b_j} = [\bar{b_j}; \tilde{b_j}; \bar{b_j} - \tilde{b_j}; \bar{b_j} \odot \tilde{b_j}], \forall j \in \{1, .., m\}$$

The authors of ESIM explain that they *"expect [...] such operations [to] help sharpen local inference information between elements in the tuples and capture inference relationships such as contradiction"* (Chen, Zhu, Ling, Wei, et al., 2017a).

3.1.3 Inference Composition

To compose the information captured by the c_{a_i} and c_{b_j}, a second bi-LSTM is used in the network. However, because these vectors introduce a lot of new dimensions to the problem, the c_{a_i} and c_{b_j} are first passed through a mapping function F consisting in a simple feed-forward layer with $ReLU$ activation, to control the model's complexity:

$$\bar{c}_{a_i} = F(c_{a_i}) = ReLU(W_F c_{a_i} + b_F)$$
$$\bar{c}_{b_j} = F(c_{b_j}) = ReLU(W_F c_{b_j} + b_F)$$

After passing the $\bar{c}_{a_i}$ and $\bar{c}_{b_j}$ through the second bi-LSTM, new vectors are obtained:

$$v_{a_i} = [LSTM_2(\bar{c}_{a_1}, \bar{c}_{a_2}, ..., \bar{c}_{a_i}); LSTM_2(\bar{c}_{a_l}, \bar{c}_{a_{l-1}}, ..., \bar{c}_{a_i})]$$
$$v_{b_j} = [LSTM_2(\bar{c}_{b_1}, \bar{c}_{b_2}, ..., \bar{c}_{b_j}); LSTM_2(\bar{c}_{b_m}, \bar{c}_{b_{m-1}}, ..., \bar{c}_{b_j})]$$

Then, to merge the v_{a_i} and the v_{b_j}, average and max. pooling operations are applied on vectors belonging to the same sentence, and the results are concatenated in a final representation v:

$$v_{a,avg} = \sum_{i=1}^{l} \frac{v_{a_i}}{l}, \qquad v_{a,max} = \max_{i=1}^{l} v_{a_i}$$
$$v_{b,avg} = \sum_{j=1}^{m} \frac{v_{b_j}}{m}, \qquad v_{b,max} = \max_{j=1}^{m} v_{b_j}$$

$$v = [v_{a,avg}; v_{a,max}; v_{b,avg}; v_{b,max}]$$

To predict the probabilities of the classes associated to the input premise and hypothesis, the vector v is passed through a two-layer perceptron G with *tanh* and *softmax* activation functions:

$$y = G(v) = softmax(W_{G_2} tanh(W_{G_1} v + b_{G_1}) + b_{G_2})$$

The class predicted by the model is the one with the highest probability in y.

3.2 Implementation with PyTorch

To evaluate our baseline and later enhance it with LE information (more on that in chapter 4), we implemented ESIM with *PyTorch*[2], an efficient and easy to use *Python* framework for deep learning. Our code is publicly available on Github in the repository at the address `https://github.com/coetaur0/ESIM`.

3.2.1 Structure of the Code

The code in our repository is organised as follows:

- The actual implementation of the model is located in the *esim/* folder, which acts as in installable Python package that can be used both to pre-process NLI data and to create an instance of our implementation of ESIM that can be trained and tested with PyTorch.
- The *scripts/* folder of the repository contains Python scripts to download and pre-process the data necessary for the model (NLI data sets and pre-trained word embeddings), as well as all the code used to train and test ESIM on SNLI, MultiNLI and the *Breaking NLI* data set.
- All the data downloaded and pre-processed with the scripts mentioned above is saved in the *data/* folder of the repository. NLI data sets are stored in the *dataset/* sub-directory after they have been downloaded, pre-trained embeddings in *embeddings/*, and pre-processed data in the *preprocessed/* folder.
  The *data/* folder is also used to save checkpoints containing the state and the weights of the model produced at each epoch during training. The checkpoints can later be used to test the model or to resume training from a certain point.
- The *config/* folder of the repository contains JSON files defining the specific parameters used to pre-process the data and to train and test ESIM. These files are called by the code in the *scripts/* folder at execution time.

3.2.2 The `esim` package

As mentioned in the previous section, the *esim/* folder of our repository acts as a Python package that can be imported in scripts to pre-process NLI data and to train and test ESIM.

In the sections below, we provide a high-level view of the organisation of the package and some information about our implementation of the model. However, we invite the reader to directly consult the sources for more details about its inner working and the optimisations we applied to speed up computations on GPUs. All the classes, methods and functions in the code are fully documented with inline Python *docstrings* to make our implementation as easy to understand as possible.

[2] `https://pytorch.org/`

3.2.2.1 Pre-processing

All the code relative to the pre-processing phase is located in the *data.py* file of the directory. After the package has been installed, the corresponding Python module can be imported in another Python file with the `import esim.data` instruction.

The `esim.data` module defines a `Preprocessor` class that can be used to open an existing NLI data set, to compute its *word dictionary* (a dictionary associating each word in the data set's vocabulary to a unique index) and to transform the words in the sentences that compose it to their indices. The class can also be used to build embedding matrices with pre-trained vectors for the data set's word dictionary.
When it is instantiated, `Preprocessor` takes some parameters as input to define if words in the pre-processed data set need to be lower-cased, if punctuation should be ignored, if stop words should be removed, etc.

In addition to `Preprocessor`, the `esim.data` module also defines a class called `NLIDataset` that can be used to iterate over the instances of a pre-processed data set. `NLIDataset` inherits from PyTorch's `Dataset` class, which is designed to be used with the framework's special `DataLoader` iterator class. This class allows users to easily read data in batches and to shuffle it, which is particularly useful during the training of deep learning models.

3.2.2.2 Model

The code for the implementation of ESIM is spread over multiple files in the *esim/* folder:

- In the *utils.py* file, all the low-level functions used in the model to perform special computations on batches of data, such as masked operations, are defined. Batched operations are used in our implementation to optimise its performance on GPUs and shorten training and testing times.

- In *layers.py*, the different custom layers needed by ESIM are defined.
  In particular, we implement a special *sequence-to-sequence encoder* in the `Seq2SeqEncoder` class, because PyTorch lacks support for processing batches of variables length sequences with recurrent neural networks.
  We also define a `RNNDropout` layer to correctly apply dropout on the input of RNNs (in the case of RNNs, dropout must be applied on the same dimensions of each input vector of the same sequence, which cannot be done with PyTorch's default `Dropout` module).
  Finally, we define the layer to compute attention between words in premises and hypotheses in the `SoftmaxAttention` class of the module.

- The actual definition of the architecture of ESIM is described in the *model.py* file of the package, in the `ESIM` class. The class inherits from `torch.nn.Module`, which is used in PyTorch to define layers or models that can be trained with the framework's special *autograd*[3] mechanism.

[3] `https://pytorch.org/docs/stable/notes/autograd.html`

The `ESIM` class simply stacks the custom layers from *layers.py* with other predefined PyTorch modules to build the model's architecture as it was described in section 3.1.
The model can be used in other Python modules or script by importing it with the instruction `from esim.model import ESIM`.

3.2.3 Scripts

Different scripts are used in the repository to perform operations with the classes from the `esim` package:

- The *fetch_data.py* script can be used to download NLI data sets such as SNLI and pre-trained embeddings like *GloVe* so that they can later be used in the model.

- The *preprocessing/* sub-directory of the *scripts/* folder contains code to preprocess SNLI, MultiNLI and the *Breaking NLI* data set with the `Preprocessor` class from `esim.data`.
  The scripts take configurations file as argument to define if words in the data sets need to be lower-cased, stop words removed, punctuation ignored, and if beginning- and end-of-sentence tokens must be used. By default, the parameters from the files in the *config/preprocessing* folder of the repository are used.

- In the *training/* sub-folder, the scripts to train ESIM on SNLI and MultiNLI are defined. Again, they take configuration files as argument to determine their training parameters, such as the number of epochs to apply, the batch size to use, the optimiser's learning rate, etc.. Default configuration files are located in the *config/training* directory.

- The scripts to test pre-trained versions of ESIM on SNLI, MultiNLI and the *Breaking NLI* data set are located in the *testing/* sub-folder of *scripts/*.
  The same file can be used to test the model on both SNLI and *Breaking NLI*: the *test_snli.py* script must be called with the paths to a pre-processed test set and a checkpoint as arguments to compute ESIM's accuracy on the selected data set.
  In the case of MultiNLI, the *test_mnli.py* script is called with a configuration file and the path to a checkpoint. A default configuration is provided in the *config/testing* folder of the repository.

All the instructions to install the `esim` package and its dependencies on a machine and to use the scripts to train and test the model are given in the repository's `README`.

A checkpoint containing the weights of the best model we trained with our implementation on SNLI is also provided in the *checkpoints/SNLI/* folder. This checkpoint can be used with the *test_snli.py* script to reproduce our results without having to train the entire model again.

3.3 Evaluation and Results

3.3.1 Data Sets and Evaluation Procedure

In order to evaluate the performance of our implementation of ESIM, we train and test the model on three famous NLI tasks: SNLI, MultiNLI and the *Breaking NLI* data set.

For SNLI, the model is first trained on the corpus' training set and validated on its development set. Then, the weights of the model at the epoch were it performs best on the development set are re-used to evaluate it on the test set, and its classification accuracy is reported (the accuracy is the percentage of correct predictions returned by the model).

The same procedure is followed to evaluate ESIM on MultiNLI, with the difference that the labels predicted by the model for the instances in the *matched* and *mismatched* test sets of MultiNLI have to be submitted to the data set's *Kaggle* competitions[4][5] to retrieve its classification accuracy on them.

As reported in section 2.1.1.3, the *Breaking NLI* (BNLI) data set consists only in an additional test set for SNLI. Hence, the weights of the best model pre-trained on SNLI are re-used to evaluate it on BNLI.

3.3.2 Training Details and Parameters

The exact same procedure and parameters as the ones reported in the paper by Chen, Zhu, Ling, Wei, et al. (2017a) are used to train our implementation of ESIM:

- During pre-processing, *beginning-* and *end-of-sentence* (`BOS` and `EOS`) tokens are added at the beginning and end of each sentence in the data sets. All words in the data sets' sentences are simply tokenised, without lemmatising or lowercasing them, and stop words are kept.
- The embeddings for out-of-vocabulary words are initialised randomly following a normal distribution, and all embeddings are updated during training.
- To optimise the model's weights, the *Adam* learning algorithm is used with an initial learning rate of 0.0004, a first momentum of 0.9, a second momentum of 0.999 and batches of 32 instances.
- Dropout with a rate of 0.5 is applied on all feed-forward connections in the network during training. Gradient clipping is also used with a maximum value of 10.0.
- The size of all hidden layers in the network is set to 300.

[4] `https://www.kaggle.com/c/multinli-matched-open-evaluation`
[5] `https://www.kaggle.com/c/multinli-mismatched-open-evaluation`

- A maximum of 64 epochs are applied to train the model. Early stopping with a patience of 5 is used: if after 5 epochs the classification accuracy of the model on the validation set doesn't improve, training is stopped.

3.3.3 Results

The classification accuracy of our implementation of ESIM on SNLI is reported in table 3.1 below. The results are in line with those reported by Chen, Zhu, Ling, Wei, et al. (2017a).

Split	**Accuracy (%)**
Training	93.2
Development	88.4
Testing	88.0

Table 3.1: Accuracy of our implementation of ESIM on the SNLI corpus

On MultiNLI, our implementation obtains the results presented in table 3.2. The values are slightly above those reported by Williams et al. (2018).

Split	**Accuracy (%)**
Training	87.7
Development matched	77.0
Development mismatched	76.8
Testing matched	76.6
Testing mismatched	75.8

Table 3.2: Accuracy of our implementation of ESIM on the MultiNLI corpus

Finally, on the *Breaking NLI* data set, the model pre-trained on SNLI reaches an accuracy of **65.5%**, which is similar to the value reported by Glockner et al. (2018) for their implementation.

Chapter 4

Lexical Entailment Augmented Network

In this chapter, we introduce the *Lexical Entailment Augmented Network* (LEAN), a model for NLI based on the architecture of ESIM, but enhanced with lexical entailment information about the words it receives as input.

During the building of LEAN, three specific LE metrics were used to try and enhance the model. These are presented in section 4.1 of this chapter.
Multiple ways of including the LE metrics in the model were also investigated and are described in section 4.2.
In section 4.3, our implementation of LEAN with PyTorch is briefly introduced.
Finally, the procedure and parameters used to train and test the model are reported in section 4.4, as well as the results obtained with LEAN on the three data sets used to evaluate ESIM in the previous chapter.

4.1 Lexical Entailment Metrics

To investigate the use of lexical entailment in NLI, three LE metrics are integrated into LEAN to see if they can help it predict inference. These metrics, which were already introduced in section 2.2.3, are detailed in the following sub-sections.

4.1.1 Word2Hyp

The first metric that we include in LEAN is the one associated with the *Word2Hyp* word embeddings presented by Henderson (2017). To compute it, the words in the premises and hypotheses of NLI data sets are transformed to their respective *Word2Hyp* embeddings, and the formula presented in section 2.2.3.1 is then used to determine if they entail each other or not:

$$log(P(x \Rightarrow y)) \approx X \oslash Y = \sigma(-X) \cdot log(\sigma(-Y))$$

In the formula, X and Y are the embeddings for words x and y, and σ is the *sigmoid* function. With this metric, higher values indicate a relation of LE between

two words, while smaller values mean that no such relation exists between them.

Pre-trained word embeddings provided by Dr. James Henderson are used for this part, and vectors for words absent from the *Word2Hyp* vocabulary are initialised randomly as follows:

- First, for each dimension of the pre-trained vectors, the mean and standard deviation are computed over all available embeddings.
- Then, vectors for unseen words are initialised randomly, dimension by dimension, following normal distributions with the means and standard deviations computed in the previous step.

4.1.2 LEAR

The second metric that we use in LEAN is the one defined for the LEAR word embeddings (Vulić and Mrkšić, 2018). Again, to compute it, words in premises and hypotheses are transformed to their respective LEAR vectors, and the degree of entailment between them is measured with the formula from section 2.2.3.2:

$$I_{LE}(x, y) = dcos(x, y) + D_2(x, y)$$

where $dcos(x, y)$ is the cosine distance between embeddings x and y, and $D_2(x, y)$ is a measure of the difference between their norms, defined as follows:

$$D_2(x, y) = \frac{||x|| - ||y||}{||x|| + ||y||}$$

where $||x||$ and $||y||$ denote the euclidean norms of x and y, respectively.

In the case of the I_{LE} metric, since it is a distance, smaller values indicate a higher degree of entailment between two words, and larger values a weaker form of entailment or no entailment at all.

For all our measures, we use the pre-trained word embeddings distributed by the authors of LEAR on their Github repository[1]. Unseen words are initialised randomly in the same manner as *Word2Hyp* vectors.

4.1.3 SDSN

In addition to *Word2Hyp* and LEAR, we also tried to integrate the measures of LE produced by the SDSN (Rei et al., 2018) into LEAN during our experiments.

The original implementation of the model, available on GitHub[2], was done with *Theano*[3], a deep learning framework for Python. Unfortunately, the code in that repository didn't work well with our version of LEAN. Hence, we reproduced the

[1] `https://github.com/nmrksic/LEAR`
[2] `https://github.com/marekrei/sdsn`
[3] `http://deeplearning.net/software/theano/`

SDSN with PyTorch to include it in our computations, and we re-used the pre-trained weights made available by its authors on their GitHub repository to make predictions with it.

The experiments we conducted with the SDSN showed that including it in LEAN did not improve the model's performance, and on the contrary even deteriorated it. This is the reason why this metric was eventually not included into our model and is absent from the rest of this chapter.

4.2 Inclusion of Lexical Entailment in LEAN

Three main approaches were investigated in this work for the integration of the two first metrics from section 4.1 into LEAN. The following sub-sections detail them.

4.2.1 First Approach: LEAN-1

A high-level view of the first approach evaluated in this work is represented in figure 4.1. The model, referred to as LEAN-1 throughout the rest of this document, essentially follows the same architecture as ESIM, only with the addition of two new components (circled in red in the figure).

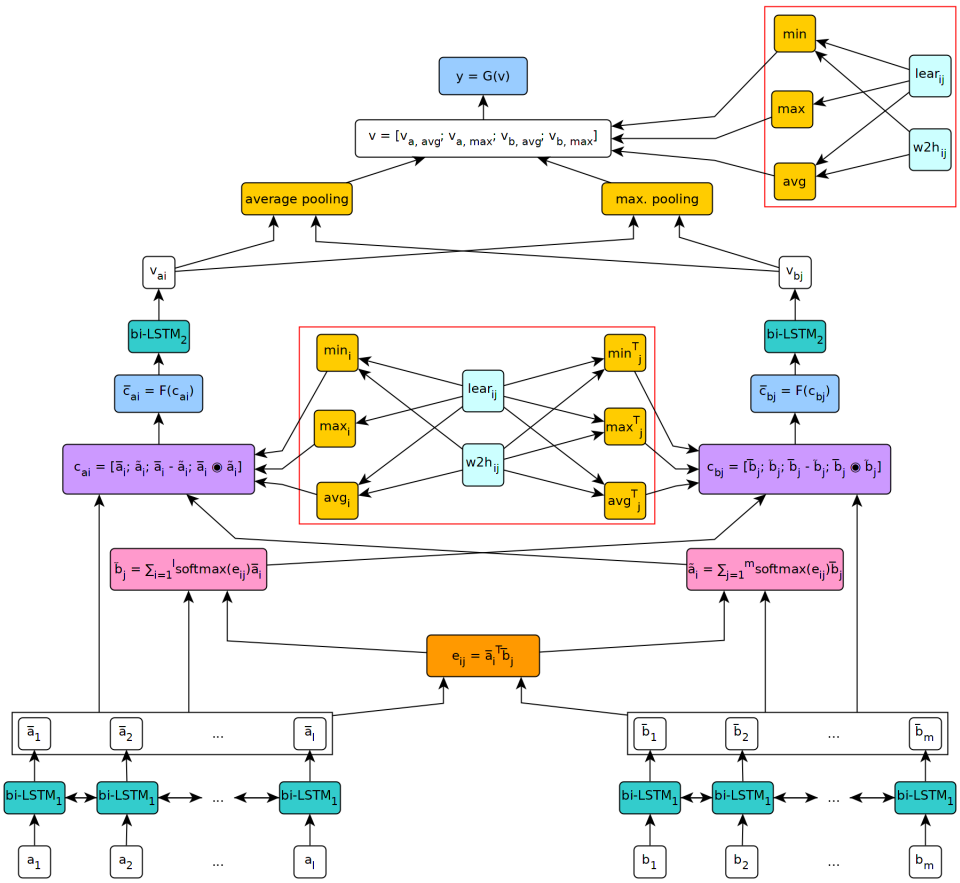

Figure 4.1: Architecture of LEAN-1

The new components add supplementary lexical entailment features to the c_{a_i}, c_{b_j} and v vectors of ESIM. In the figure, $w2h_{ij}$ and $lear_{ij}$ are matrices containing the lexical entailment scores from sections 4.1.1 and 4.1.2 computed between the words in the premise and hypothesis of a sentence pair. A conceptual representation of how these matrices are produced for two sentences $a = a_i, \forall i \in \{1, ..., l\}$ and $b = b_j, \forall j \in \{1, ..., m\}$ is given in figure 4.2.

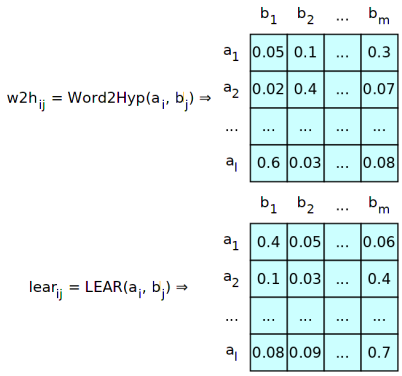

Figure 4.2: Lexical entailment matrices in LEAN-1

4.2.1.1 First Component

For each word in the premise of a sentence pair, the average of the lexical entailment scores between that word and all of those in the hypothesis is computed, as well as the maximum and the minimum value (this step is represented by the avg_i, max_i and min_i nodes in figure 4.1). This procedure is applied on the two lexical entailment matrices used in LEAN, which produces six new features per word in the premise that are concatenated to the c_{a_i} vectors. The same is done the other way around for every word in the hypothesis (with the operations represented by avg_j^T, max_j^T and min_j^T), and the additional features are appended to the vectors c_{b_j}.

Figure 4.3 below illustrates how the avg_i and max_i operations are computed, and figure 4.4 what the transpose in avg_j^T means in practice.

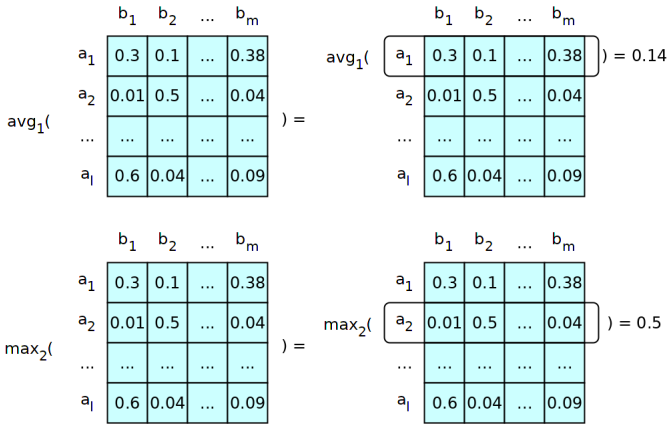

Figure 4.3: Computation of the word-level average and maximum LE score

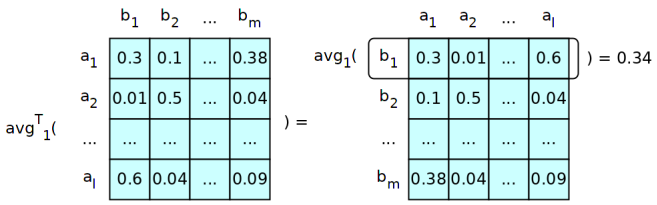

Figure 4.4: Computation of the transposed word-level average LE score

The objective of this step is to try and enhance the local inference modelling layer of ESIM with additional lexical entailment information explicitly computed between all the words in a sentence pair.

4.2.1.2 Second Component

In the second component, the average, maximum and minimum lexical entailment scores are computed at the sentence level for each LE metric and concatenated to the vector v. In practice, this is done by computing the average, maximum and minimum LE scores over the entire lexical entailment matrices instead of on only one row or column like in the first component of LEAN-1.

The objective of this step is to try and capture more general information about lexical entailment between the two sentences in a pair, and to use it to augment ESIM's inference composition layer.

4.2.2 Second Approach: LEAN-2

If we look into the words that compose the vocabularies of the two LE metrics used in LEAN-1, we observe that they are mostly nouns and verbs, and that there are almost no "structural" or *stop words* among them. This is because stop words don't usually carry much information about lexical entailment: they participate more to the syntax of a sentence than to its semantics.

Since they are absent from the vocabularies of the LE metrics, embeddings for stop words are randomly generated in LEAN-1, and the scores obtained with them do not really reflect entailment. Because of this, it seems logical not to include these words into the computation of lexical entailment between premises and hypotheses. However, in LEAN-1, the first component appends LE features to the vectors c_{a_i} and c_{b_j} of every word in the premise and hypothesis, including stop words. This means that not computing lexical entailment for them would force us to completely remove them from the sentences passed as input to the model. This approach seems too restrictive, as structural words might still carry information about syntax that could be of interest for the other components of LEAN.

The solution we propose in LEAN-2 is to limit our computation of lexical entailment scores to words that aren't stop words, like nouns and verbs, but to only use the second component from LEAN-1 to include that information in the model. This means that we only compute the average, maximum and minimum lexical entail-

ment scores at the sentence level for a pair, and not for every word like in LEAN-1, which allows us to keep stop words in the rest of the model.

4.2.3 Third Approach: LEAN-3

The third and last approach evaluated in this work, called LEAN-3, investigates whether the inclusion of lexical entailment in the computation of soft attention improves the model's performance. To do so, the LEAR and *Word2Hyp* LE matrices are used in a weighted sum with the attention matrix e_{ij} from ESIM:

$$s_{ij} = e_{ij} + \lambda_1 w2h_{ij} + \lambda_2 lear_{ij}$$

where λ_1 and λ_2 are parameters learned during training. The resulting matrix s_{ij} is then used to compute attention between words in premises and hypotheses in the same way that e_{ij} is used in ESIM.
The rest of the model follows LEAN-1's architecture.

Figure 4.5 below illustrates a high-level view of LEAN-3.

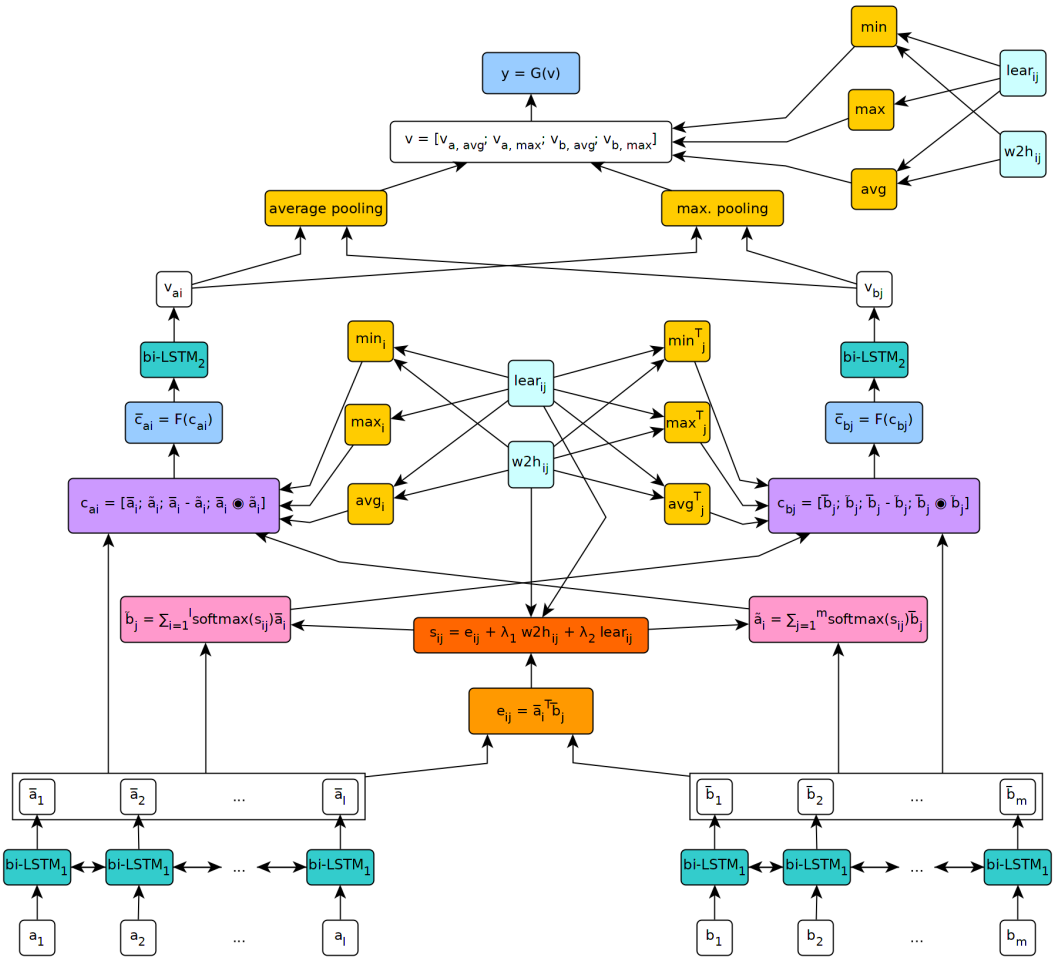

Figure 4.5: Architecture of LEAN-3

4.2.4 Other Approaches

Aside from LEAN-1, 2 and 3, we also tried in earlier iterations to just replace the attention mechanism from ESIM with one that only used the *Word2Hyp* and LEAR lexical entailment matrices.

To determine if "hard" decisions given by binary lexical entailment scores were more informative than "soft" ones with continuous values, we also tried to binarise the LE matrices before using them in all three versions of LEAN. To do so, we applied thresholds on their values to determine whether there was a relation of entailment between two words (1), or not (0):

$$w2h_{ij}^{b} = \begin{cases} 1, & \text{if } w2h_{ij} > t^{w2h} \\ 0, & \text{otherwise} \end{cases}, \forall i, j$$

$$lear_{ij}^{b} = \begin{cases} 1, & \text{if } lear_{ij} > t^{lear} \\ 0, & \text{otherwise} \end{cases}, \forall i, j$$

In the formulas above, t^{w2h} and t^{lear} are thresholds defined as the medians of all values computed for each lexical entailment metric on the whole training set of a NLI corpus.

Our experiments with the two approaches described in this section showed that they didn't improve LEAN's performance, which is why they were eventually abandoned.

4.3 Implementation with PyTorch

Our code for LEAN is publicly available on GitHub at the address `https://github.com/coetaur0/LEAN`. The repository is organised in the exact same way as the one from section 3.2, except that the *esim/* folder is replaced by a *lean/* directory. The main differences with the implementation of ESIM are the following:

- In *utils.py*, additional functions to compute lexical entailment between pre-trained *Word2Hyp* and LEAR embeddings are defined, as well as functions to compute the masked average, maximum and minimum on batches of lexcial entailment matrices.
- The *lean.py* file replaces *esim.py* and defines three models for the different versions of LEAN instead of one for ESIM.
- An additional *sdsn.py* file contains an implementation with PyTorch of the SDSN to compute entailment on sentence pairs with the metric.

For more details about the implementation of LEAN with PyTorch, we invite the reader to directly consult the code in the repository. Again, all the classes, methods and functions are fully documented with inline Python *docstrings*, and instructions on how to train and test the model are given in the repository's README.

4.4 Evaluation and Results

4.4.1 Data Sets and Evaluation Procedure

In order to compare it with ESIM and to determine if lexical entailment information does indeed improve the performance of the model, LEAN is trained and tested on the same three data sets as ESIM, following the exact same procedure.

4.4.2 Training Parameters

The same parameters as those used for ESIM are applied on LEAN during training, only with the following small differences in the pre-processing phase:

- In LEAN-1 and LEAN-3, all words are lowercased during pre-processing, and no beginning- or end-of-sentence tokens are added to the premises and hypotheses. This choice is made because the vocabularies of the LE metrics do not contain embeddings for uppercased words or for `BOS` and `EOS`.
- In LEAN-2, the same parameters as in LEAN-1 and 3 are used, but stop words are removed from sentences when lexical entailment is computed between them.

During the training phase, we also performed some hyper-parameter optimisation on the layers where lexical entailment was added to fine-tune our models and get the best performance possible.

4.4.3 Results

Table 4.1 summarises the accuracies obtained with each version of LEAN on the SNLI corpus. Values in parentheses indicate the difference with the results obtained with ESIM on the same corpus.

Split	**LEAN-1**	**LEAN-2**	**LEAN-3**
Training	93.3%	93.0%	93.0%
Development	88.5%	88.4%	88.2%
Testing	**88.4% (+0.4)**	88.1% (+0.1)	88.2% (+0.2)

Table 4.1: Accuracy of the three versions of LEAN on SNLI

The results obtained with LEAN on MultiNLI are reported in table 4.2.

Split	**LEAN-1**	**LEAN-2**	**LEAN-3**
Training	88.3%	88.9%	89.6%
Development matched	77.6%	77.0%	77.7%
Development mismatched	77.6%	77.4%	77.6%
Testing matched	77.7% (+1.1)	77.3% (+0.7)	**78.1% (+1.5)**
Testing mismatched	**76.8% (+1.0)**	76.0% (+0.2)	76.2% (+0.4)

Table 4.2: Accuracy of the three versions of LEAN on MultiNLI

Finally, table 4.3 presents the results obtained on the *Breaking NLI* data set.

Split	**LEAN-1**	**LEAN-2**	**LEAN-3**
Testing	**62.6% (–2.9)**	60.7% (–4.8)	61.8% (–3.7)

Table 4.3: Accuracy of the three versions of LEAN on the *Breaking NLI* data set

Chapter 5

Discussion of the Results

5.1 Comparison between LEAN and ESIM

The results from section 4.4.3 show that the inclusion of lexical entailment in LEAN does help it predict inference more accurately than ESIM. However, the reported increases in performance aren't as important as our initial intuition led us to believe they would be (especially in the case of SNLI), and we even observe a decrease in accuracy on the *Breaking NLI* data set.

There are a number of potential explanations for this. First of all, during the pre-processing phase of our experiments, when we build the embedding matrices for the LE metrics used in LEAN, we count the number of words absent from their vocabularies and initialised randomly. On a total of 33,250 different words in SNLI, 8,131 are absent from the pre-trained embeddings for LEAR, while that number goes as far up as 11,749 for *Word2Hyp*.
Although we try to produce vectors as close as possible to the spaces of the metrics during their random initialisation, the embeddings for unseen words do not truly carry information about lexical entailment. This means that the inclusion of LE scores computed with them in LEAN potentially adds a lot of noise in the model, which prevents it from performing as well as if it had access to real pre-trained vectors for its entire vocabulary.

On the *Breaking NLI* data set, this phenomenon most likely has an even bigger impact than on SNLI, because premises and hypotheses only differ by single words. Hence, if the word being replaced in a premise or its substitute in the hypothesis is absent from the LE metrics vocabularies, only noise is added in LEAN, instead of true lexical entailment information. In that situation, the inclusion of the metrics in the model probably does more harm than good, which might explain why LEAN's performance on *Breaking NLI* is lower than ESIM's.

In the case of MultiNLI, even more words are absent from the LE metrics' vocabularies than is the case for SNLI (28,508 for *Word2Hyp* and 17,717 for LEAR, on a total of 70,523 words in the data set). However, LEAN's increase in performance is larger on MultiNLI than on SNLI. The elements mentioned above can therefore not constitute the sole explanation for the relatively small improvements between

our model and the baseline.

This observation leads us to our second hypothesis as to why our model's performance isn't as good as we initially thought it would be: the metrics we had access to and included in the model aren't as informative on NLI as we thought. Indeed, both *Word2Hyp* and LEAR are measures of lexical entailment in its more restrictive sense of hyponymy/hypernymy. Hence, they are much more limited than true LE and they do not model the other lexical semantic relations that it covers, such as synonymy or meronymy.

If we pick random examples of sentence pairs connected through entailment in SNLI, like the ones in table 5.1 below, we observe that in most cases hypernymy plays little to no role in the decision whether a premise entails a hypothesis or not. Indeed, as the examples show, a lot of the pairs in the NLI data set consist in sentences containing the same words but organised differently, or in one sentence that is a subset of the other. Only rarely does a relation of hypernymy between words in a premise and hypothesis need to be captured to properly recognise entailment. The last sentence in table 5.1 is one of the few examples that illustrate such a situation: because the word *ladies* entails *women*, the hypothesis can be inferred from the premise.

Premise	**Hypothesis**
Chevrolet car on display at a convention.	At a convention, there is a chevrolet car on display.
Several runners compete in a road race.	Several runners compete in a race.
Two ladies looking for bread to purchase.	Two women looking at bread.

Table 5.1: Examples of sentence pairs labelled with *entailment* in SNLI

The other problem with the inclusion of measures of hypernymy in LEAN is that they are mostly informative about the relation of textual entailment, and not so much about contradiction or neutral pairs. The lexical entailment metrics used in the model hence probably only help it in a smaller number of cases than we initially thought, which could be why its increase in performance is not that large compared to other approaches that also use other lexical relations, like KIM (Chen, Zhu, Ling, Inkpen, et al., 2018).

Despite all of the elements mentioned above, we still observe an interesting increase in classification accuracy between ESIM and LEAN. The results seem to indicate that lexical entailment does provide our model with information that helps it better predict inference.

Although the improvements on SNLI are not as good as we expected, in the case of MultiNLI the difference between the baseline and our proposed model is actually quite significant (more than +1% accuracy on both the matched and mismatched test sets), which indicates that lexical entailment is probably more informative for that data set. Our model even outperforms KIM by 0.5% on the matched test set and 0.4% on the mismatched version.

It is possible that the difference with SNLI could be attributed to the wider range of types of text covered by MultiNLI: maybe the larger variety of topics addressed in the corpus incurs the use of a richer vocabulary in sentence pairs and the presence of more lexical semantic relations such as hypernymy between words?

In terms of training and execution times, the cost of including lexical entailment in LEAN is nearly free. All our experiments in this work were led on a machine with an Intel i5 8600K CPU, 16GB of RAM and a NVidia GeForce GTX 1080. All the models were trained and tested on the machine's GPU. On average, it took about 17 minutes to train ESIM for one epoch, and 18 to train LEAN. In both cases, 20 epochs were usually executed before early stopping interrupted the process, which means that on average 5.7 hours were necessary to train ESIM, against 6 for LEAN. This corresponds to a difference of 18 minutes, which is more or less equivalent to training LEAN for one more epoch than ESIM.

5.2 Comparison between the Versions of LEAN

Surprisingly, the most naive of the three approaches to integrate lexical entailment in LEAN is also the most efficient. The results from section 4.4.3 show that LEAN-2 doesn't perform as well as LEAN-1, even though the model tries to remove some of the noise from LE metrics by ignoring stop words. We suspect that the main reason for this is that the first component from LEAN-1 is more important in helping the model predict inference than the second one included in LEAN-2. We will see in the ablation analysis from section 5.3 that this intuition appears to be well founded.

The lower accuracy obtained with LEAN-3 compared to the model's first version also indicates that using lexical entailment in the computation of soft attention between premises and hypotheses doesn't help the system to better capture relations between their words.
When we compare the similarity matrices from ESIM with LEAR and *Word2Hyp*'s entailment matrices, we realise that they mostly focus on the same words, only with some additional noise due to unseen words in the LE metrics. Lexical entailment hence probably doesn't add much useful information to the computation of attention, and on the contrary even slightly deteriorates the quality of the weights computed with the approach because of the noise it contains.

5.3 Analysis of the Best Version of LEAN

In this section, we perform a small ablation analysis on LEAN-1 to determine which components and which metric contribute most to the model's performance.

5.3.1 Lexical Entailment Metrics

First of all, to assess which one of *Word2Hyp* or LEAR is the most informative for LEAN, we train and test the model with only one of the metrics included in it at a

time, and we measure its accuracy on SNLI. The results are presented in table 5.2.

Model	**Train**	**Development**	**Test**
LEAN-1, *Word2Hyp* only	93.2%	88.4%	**88.2%**
LEAN-1, LEAR only	93.4%	88.3%	88.1%

Table 5.2: Ablation analysis on the LE metrics: classification accuracy on SNLI

As we can see in the table, the *Word2Hyp* metric is apparently a little more useful than LEAR in predicting inference, even though its vocabulary contains more unseen words. A potential explanation for this might be that the metric is a little more robust against noise, because it was trained in an unsupervised manner.

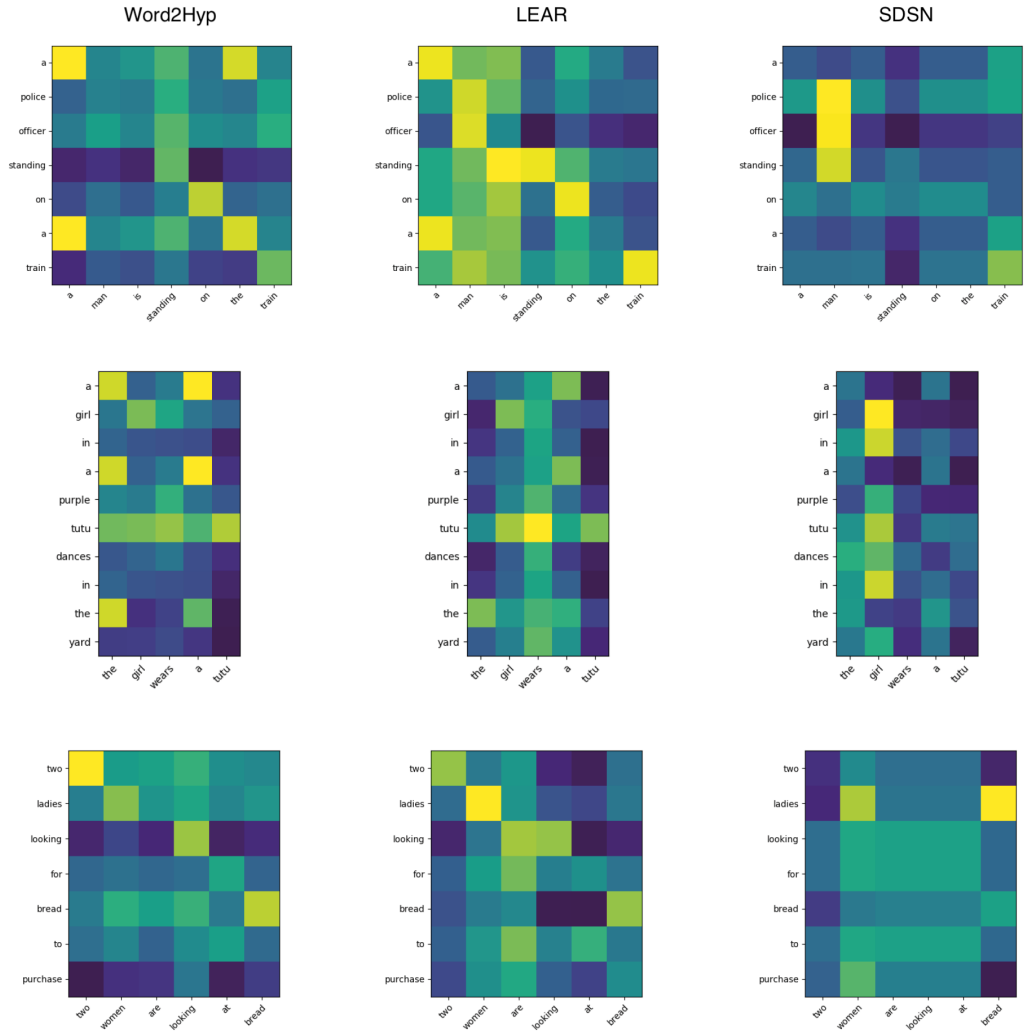

Figure 5.1: Examples of lexical entailment matrices computed with *Word2Hyp*, LEAR and the SDSN

We noted in section 4.1.3 that the inclusion of the SDSN in LEAN didn't improve the model's accuracy. The issue of overlap between vocabularies mentioned in section 5.1 is most likely the reason why.
In the SDSN, word embeddings for unseen words are not initialised randomly like in LEAR or *Word2Hyp*, because the model uses a fixed architecture with pre-trained

weights. Hence, words in SNLI or MultiNLI's vocabulary that are absent from the SDSN are all converted to a special *out-of-vocabulary* token and processed as the same vector during the computation of lexical entailment. This produces inconsistent scores that could explain why the SDSN does not help LEAN to predict inference more accurately.

Figure 5.1 presents a few examples of heat maps for lexical entailment matrices computed with *Word2Hyp*, LEAR and the SDSN. In the images, warmer colours indicate a high degree of entailment, and colder ones the opposite.
When we compare the three different approaches, it becomes quite obvious that the scores computed with the SDSN are rather inconsistent. If we look at the top three LE matrices, for example, we see that all three metrics successfully predict entailment between *police officer* and *man*, but that the SDSN is the only one that fails to recognise entailment between the other words in the sentences.
The same observation can also be made on the other examples in the figure. We remark in particular that the SDSN does not associate pairs composed of the same word twice with high entailment scores. This is problematic, since we have seen that most examples of textual entailment are composed of sentences with high word overlaps, and that recognising word similarity is therefore as important, if not more, as recognising hypernymy.

The heat maps from figure 5.1 also provide us with interesting information about the relations of entailment captured by *Word2Hyp* and LEAR. We remark for example that the two metrics are able to capture rather subtle relations between words, like the fact that *officer* entails *standing* in the first *Word2Hyp* matrix (which seems quite coherent with how we usually imagine police officers), or the word *tutu* entailing *wearing* in both metrics (again, this is quite coherent, as a tutu is a piece of clothing).

5.3.2 Components

To determine which of the two lexical entailment components in LEAN-1 participates most to its performance increase, we train and test the model with only one of them included in it at a time, and we report its classification accuracy on SNLI in table 5.3.

Model	**Train**	**Development**	**Test**
LEAN-1, 1^{st} component only	93.3%	88.5%	**88.3%**
LEAN-1, 2^{nd} component only	93.1%	88.3%	88.1%

Table 5.3: Ablation analysis on the components: classification accuracy on SNLI

The results indicate that the first component is more informative than the second one. This is probably because it provides the model with more fine-grained information about entailment between each pair of words in the premise and hypothesis. This observation is most likely the reason why LEAN-2 did not perform as well as LEAN-1, despite our attempt to remove noise from it by excluding stop words from the computation of lexical entailment scores.

5.3.3 Average, Maximum and Minimum Scores

In the last part of our ablation analysis, we investigate which of the operations used to aggregate lexical entailment scores is the most useful for LEAN-1. To do so, we train and test two versions of the model on SNLI: in the first one, only the average of the lexical entailment scores is computed, and in the second, only the maximum and minimum scores are used. The results are reported in table 5.4.

Model	**Train**	**Development**	**Test**
LEAN-1, average only	93.1%	88.4%	**88.3%**
LEAN-1, max./min. only	93.0%	88.4%	88.2%

Table 5.4: Ablation analysis on aggregation operations: classification accuracy on SNLI

The higher accuracy obtained with the average of the scores indicates that this operation is more informative to LEAN than the maximum/minimum. This makes sense, since the average actually includes the maximum and minimum in its computation, while the contrary is not true. The operation also captures information about every element in a lexical entailment matrix, and not just the ones with the highest or lowest score: if there is an outlier in the data because of noise, the maximum or minimum scores might miss useful information by solely focusing on it, while the average won't suffer as much from its effect.

Chapter 6

Conclusion

In this work, we investigated whether the use of lexical entailment metrics in an existing NLI model helped it to better model inference between natural language sentences.

In the first chapter of this document, the crux of the problem of natural language inference was explained in detail, and the importance of the task was made clear through examples of applications where NLI is useful or even necessary.

Then, in chapter 2, we explored previous work on natural language inference and lexical entailment. In both cases, we first introduced the tasks and data sets used to train and evaluate models designed to tackle the two problems. We then went on to list and describe existing approaches to perform NLI or recognise lexical entailment. In the case of natural language inference, we specifically focused on deep learning models, while we limited ourselves to the metrics considered in our proposed model for lexical entailment.
We particularly underlined in this chapter the lack of existing work investigating the use of lexical level information in NLI models, which justified our approach in this paper.

In chapter 3, we presented the model we selected as a baseline in this work and against which we compared our proposed model. We first described its architecture in detail, and we then went on to introduce our implementation for it with Pytorch. Finally, we detailed the procedure we followed to evaluate it and the parameters we used to train it, and we reported its classification accuracy on three of the most famous benchmarks for NLI.

In chapter 4, we introduced our proposed model for natural language inference, the *Lexical Entailment Augmented Network* (LEAN). We first described the lexical entailment metrics we used to augment it, and we explained the different approaches we took to include them in it. Then, we presented our implementation for it, and we reported its performance on the data sets used to evaluate the baseline.

Finally, in chapter 5, we proposed a discussion of the results obtained with LEAN. We first compared them with the baseline and tried to explain them. Then, we compared the different approaches taken in LEAN to include lexical entailment. At last, we proposed a small ablation analysis on the best version of LEAN to try

and determine which aspects of the LE metrics contributed most to our model's performance.

The results we obtained with LEAN in this work show that lexical entailment does help natural language inference models to better predict entailment between sentences. Although the increase in performance of our model compared to the baseline was not as important as we initially thought on SNLI or the *Breaking NLI* data set, we saw that the accuracy of our model on MultiNLI was still quite promising and showed that lexical entailment constituted information that was useful for inference at the sentence level.

All of these elements lead us to believe that further work should try to include more lexical level information into existing NLI models. In particular, it would be interesting to investigate whether LE can help more recent models like BERT to predict textual entailment more accurately.
More generally, our experiments in this work show that the inclusion of external information and priors into deep learning models is most likely beneficial and should hence be considered more often.

Bibliography

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR* abs/1409.0473. arXiv: 1409.0473. URL: http://arxiv.org/abs/1409.0473.

Baroni, Marco and Alessandro Lenci (2011). "How We BLESSed Distributional Semantic Evaluation". In: *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*. GEMS '11. Edinburgh, Scotland: Association for Computational Linguistics, pp. 1–10. ISBN: 978-1-937284-16-9. URL: http://dl.acm.org/citation.cfm?id=2140490.2140491.

Bowman, Samuel R., Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts (2016). "A Fast Unified Model for Parsing and Sentence Understanding". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 1466–1477. DOI: 10.18653/v1/P16-1139. URL: https://www.aclweb.org/anthology/P16-1139.

Bowman, Samuel, Gabor Angeli, Christopher Potts, and Christopher D. Manning (2015). "A large annotated corpus for learning natural language inference". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Bromley, Jane, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah (1994). "Signature verification using a" siamese" time delay neural network". In: *Advances in neural information processing systems*, pp. 737–744.

Chen, Qian, Zhen-Hua Ling, and Xiaodan Zhu (2018). "Enhancing Sentence Embedding with Generalized Pooling". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, pp. 1815–1826. URL: https://www.aclweb.org/anthology/C18-1154.

Chen, Qian, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei (2018). "Neural Natural Language Inference Models Enhanced with External Knowledge". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 2406–2417. URL: https://www.aclweb.org/anthology/P18-1224.

Chen, Qian, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen (2017a). "Enhanced LSTM for Natural Language Inference". In: *Proceedings of*

the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vancouver, Canada: Association for Computational Linguistics, pp. 1657–1668. DOI: 10.18653/v1/P17-1152. URL: https://www.aclweb.org/anthology/P17-1152.

— (2017b). "Recurrent Neural Network-Based Sentence Encoder with Gated Attention for Natural Language Inference". In: *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 36–40. DOI: 10.18653/v1/W17-5307. URL: https://www.aclweb.org/anthology/W17-5307.

Choi, Jihun, Kang Min Yoo, and Sang-goo Lee (2017). "Unsupervised Learning of Task-Specific Tree Structures with Tree-LSTMs". In: *CoRR* abs/1707.02786. arXiv: 1707.02786. URL: http://arxiv.org/abs/1707.02786.

Condori, Roque Enrique López and Thiago Alexandre Salgueiro Pardo (2017). "Opinion summarization methods: Comparing and extending extractive and abstractive approaches". In: *Expert Systems with Applications* 78, pp. 124–134. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2017.02.006. URL: http://www.sciencedirect.com/science/article/pii/S0957417417300829.

Conneau, Alexis, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes (2017). "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 670–680. DOI: 10.18653/v1/D17-1070. URL: https://www.aclweb.org/anthology/D17-1070.

Conneau, Alexis, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov (2018). "XNLI: Evaluating Cross-lingual Sentence Representations". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics.

Dagan, Ido, Oren Glickman, and Bernardo Magnini (2006). "The PASCAL Recognising Textual Entailment Challenge". In: *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment*. Vol. 3944. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 177–190. DOI: 10.1007/11736790_9. URL: http://link.springer.com/10.1007/11736790_9.

Das, Dipanjan and André FT Martins (2007). "A survey on automatic text summarization". In: *Literature Survey for the Language and Statistics II course at CMU* 4, pp. 192–195.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805. arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

Geffet, Maayan and Ido Dagan (2005). "The Distributional Inclusion Hypotheses and Lexical Entailment". In: *Proceedings of the 43rd Annual Meeting of the As-*

sociation for Computational Linguistics (ACL'05). Ann Arbor, Michigan: Association for Computational Linguistics, pp. 107–114. DOI: 10.3115/1219840.1219854. URL: https://www.aclweb.org/anthology/P05-1014.

Ghaeini, Reza, Sadid A. Hasan, Vivek Datla, Joey Liu, Kathy Lee, Ashequl Qadir, Yuan Ling, Aaditya Prakash, Xiaoli Fern, and Oladimeji Farri (2018). "DR-BiLSTM: Dependent Reading Bidirectional LSTM for Natural Language Inference". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 1460–1469. DOI: 10.18653/v1/N18-1132. URL: https://www.aclweb.org/anthology/N18-1132.

Giampiccolo, Danilo, Bernardo Magnini, Elena Cabrio, Hoa Trang Dang, Ido Dagan, and Bill Dolan (2008). "The Fourth PASCAL Recognizing Textual Entailment Challenge". In: p. 11.

Glockner, Max, Vered Shwartz, and Yoav Goldberg (2018). "Breaking NLI Systems with Sentences that Require Simple Lexical Inferences". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 650–655. URL: https://www.aclweb.org/anthology/P18-2103.

Gong, Yichen, Heng Luo, and Jian Zhang (2018). "Natural Language Inference over Interaction Space". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=r1dHXnH6-.

Gururangan, Suchin, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith (2018). "Annotation Artifacts in Natural Language Inference Data". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 107–112. DOI: 10.18653/v1/N18-2017. URL: https://www.aclweb.org/anthology/N18-2017.

Henderson, James (2017). "Learning Word Embeddings for Hyponymy with Entailment-Based Distributional Semantics". In: *CoRR* abs/1710.02437. arXiv: 1710.02437. URL: http://arxiv.org/abs/1710.02437.

Henderson, James and Diana Popa (2016). "A Vector Space for Distributional Semantics for Entailment". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 2052–2062. DOI: 10.18653/v1/P16-1193. URL: https://www.aclweb.org/anthology/P16-1193.

Hirschman, Lynette, Marc Light, Eric Breck, and John D Burger (1999). "Deep read: A reading comprehension system". In: *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. Association for Computational Linguistics, pp. 325–332.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Comput.* 9.8, pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Im, Jinbae and Sungzoon Cho (2017). "Distance-based Self-Attention Network for Natural Language Inference". In: *CoRR* abs/1712.02047. arXiv: 1712.02047. URL: http://arxiv.org/abs/1712.02047.

Kiela, Douwe, Laura Rimell, Ivan Vulić, and Stephen Clark (2015). "Exploiting Image Generality for Lexical Entailment Detection". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China: Association for Computational Linguistics, pp. 119–124. DOI: 10.3115/v1/P15-2020. URL: https://www.aclweb.org/anthology/P15-2020.

Kim, Seonhoon, Jin-Hyuk Hong, Inho Kang, and Nojun Kwak (2018). "Semantic Sentence Matching with Densely-connected Recurrent and Co-attentive Information". In: *arXiv:1805.11360 [cs]*. arXiv: 1805.11360. URL: http://arxiv.org/abs/1805.11360.

Lin, Zhouhan, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio (2017). "A Structured Self-attentive Sentence Embedding". In: *CoRR* abs/1703.03130. arXiv: 1703.03130. URL: http://arxiv.org/abs/1703.03130.

Liu, Pengfei, Xipeng Qiu, Yaqian Zhou, Jifan Chen, and Xuanjing Huang (2016). "Modelling Interaction of Sentence Pair with Coupled-LSTMs". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 1703–1712. DOI: 10.18653/v1/D16-1176. URL: https://www.aclweb.org/anthology/D16-1176.

Liu, Yang, Chengjie Sun, Lei Lin, and Xiaolong Wang (2016). "Learning Natural Language Inference using Bidirectional LSTM model and Inner-Attention". In: *CoRR* abs/1605.09090. arXiv: 1605.09090. URL: http://arxiv.org/abs/1605.09090.

Marelli, Marco, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli (2014). "A SICK cure for the evaluation of compositional distributional semantic models". In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Reykjavik, Iceland: European Language Resources Association (ELRA), pp. 216–223. URL: http://www.lrec-conf.org/proceedings/lrec2014/pdf/363_Paper.pdf.

McCann, Bryan, James Bradbury, Caiming Xiong, and Richard Socher (2017). "Learned in Translation: Contextualized Word Vectors". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., pp. 6294–6305. URL: http://papers.nips.cc/paper/7209-learned-in-translation-contextualized-word-vectors.pdf.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Distributed Representations of Words and Phrases and Their Compositionality". In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., pp. 3111–3119. URL: http://dl.acm.org/citation.cfm?id=2999792.2999959.

Miller, George A. (1995). "WordNet: A Lexical Database for English". In: *Commun. ACM* 38.11, pp. 39–41. ISSN: 0001-0782. DOI: 10.1145/219717.219748. URL: http://doi.acm.org/10.1145/219717.219748.

Mou, Lili, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin (2015). "Recognizing Entailment and Contradiction by Tree-based Convolution". In: *CoRR* abs/1512.08422. arXiv: 1512.08422. URL: http://arxiv.org/abs/1512.08422.

Mrkšić, Nikola, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young (2017). "Semantic Specialization of Distributional Word Vector Spaces using Monolingual and Cross-Lingual Constraints". In: *Transactions of the Association for Computational Linguistics* 5, pp. 309–324. DOI: 10.1162/tacl_a_00063. URL: https://www.aclweb.org/anthology/Q17-1022.

Munkhdalai, Tsendsuren and Hong Yu (2017). "Neural Semantic Encoders". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 397–407. URL: https://www.aclweb.org/anthology/E17-1038.

Nelson, Douglas L., Cathy L. McEvoy, and Thomas A. Schreiber (2004). "The University of South Florida free association, rhyme, and word fragment norms". In: *Behavior Research Methods, Instruments, & Computers* 36.3, pp. 402–407. ISSN: 1532-5970. DOI: 10.3758/BF03195588. URL: https://doi.org/10.3758/BF03195588.

Nie, Yixin and Mohit Bansal (2017). "Shortcut-Stacked Sentence Encoders for Multi-Domain Inference". In: *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 41–45. DOI: 10.18653/v1/W17-5308. URL: https://www.aclweb.org/anthology/W17-5308.

Pan, Boyuan, Yazheng Yang, Zhou Zhao, Yueting Zhuang, Deng Cai, and Xiaofei He (2018). "Discourse Marker Augmented Network with Reinforcement Learning for Natural Language Inference". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 989–999. URL: http://aclweb.org/anthology/P18-1091.

Parikh, Ankur, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit (2016). "A Decomposable Attention Model for Natural Language Inference". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 2249–2255. DOI:

10.18653/v1/D16-1244. URL: https://www.aclweb.org/anthology/D16-1244.

Pasunuru, Ramakanth, Han Guo, and Mohit Bansal (2017). "Towards improving abstractive summarization via entailment generation". In: *Proceedings of the Workshop on New Frontiers in Summarization*, pp. 27–32.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer (2018). "Deep contextualized word representations". In: *Proc. of NAACL*.

Poliak, Adam, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme (2018). "Hypothesis Only Baselines in Natural Language Inference". In: *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 180–191. DOI: 10.18653/v1/S18-2023. URL: https://www.aclweb.org/anthology/S18-2023.

Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever (2018). "Improving language understanding by generative pre-training". In:

Rei, Marek, Daniela Gerz, and Ivan Vulic (2018). "Scoring Lexical Entailment with a Supervised Directional Similarity Network". In: *ACL*.

Rocktäschel, Tim, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom (2016). "Reasoning about Entailment with Neural Attention". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. URL: http://arxiv.org/abs/1509.06664.

Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei (2015). "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3, pp. 211–252. ISSN: 1573-1405. DOI: 10.1007/s11263-015-0816-y. URL: https://doi.org/10.1007/s11263-015-0816-y.

Sha, Lei, Baobao Chang, Zhifang Sui, and Sujian Li (2016). "Reading and thinking: Re-read LSTM unit for textual entailment recognition". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 2870–2879.

Shen, Tao, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang (2017). "DiSAN: Directional Self-Attention Network for RNN/CNN-free Language Understanding". In: *CoRR* abs/1709.04696. arXiv: 1709.04696. URL: http://arxiv.org/abs/1709.04696.

Shen, Tao, Tianyi Zhou, Guodong Long, Jing Jiang, Sen Wang, and Chengqi Zhang (2018). "Reinforced Self-Attention Network: a Hybrid of Hard and Soft Attention for Sequence Modeling". In: *CoRR* abs/1801.10296. arXiv: 1801.10296. URL: http://arxiv.org/abs/1801.10296.

Srivastava, Rupesh Kumar, Klaus Greff, and Jürgen Schmidhuber (2015). "Training Very Deep Networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Montreal, Canada: MIT Press, pp. 2377–2385. URL: http://dl.acm.org/citation.cfm?id=2969442.2969505.

Talman, Aarne, Anssi Yli-Jyrä, and Jörg Tiedemann (2018). "Natural Language Inference with Hierarchical BiLSTM Max Pooling Architecture". In: *CoRR* abs/1808.08762. arXiv: 1808.08762. URL: http://arxiv.org/abs/1808.08762.

Tay, Yi, Anh Tuan Luu, and Siu Cheung Hui (2018). "Compare, Compress and Propagate: Enhancing Neural Architectures with Alignment Factorization for Natural Language Inference". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 1565–1575. URL: https://www.aclweb.org/anthology/D18-1185.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., pp. 5998–6008. URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Vulić, Ivan, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen (2017). "HyperLex: A Large-Scale Evaluation of Graded Lexical Entailment". In: *Computational Linguistics* 43.4, pp. 781–835. DOI: 10.1162/COLI_a_00301. URL: https://www.aclweb.org/anthology/J17-4004.

Vulić, Ivan and Nikola Mrkšić (2018). "Specialising Word Vectors for Lexical Entailment". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 1134–1145. DOI: 10.18653/v1/N18-1103. URL: https://www.aclweb.org/anthology/N18-1103.

Wang, Shuohang and Jing Jiang (2016). "Learning Natural Language Inference with LSTM". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, pp. 1442–1451. DOI: 10.18653/v1/N16-1170. URL: https://www.aclweb.org/anthology/N16-1170.

Wang, Zhiguo, Wael Hamza, and Radu Florian (2017). "Bilateral Multi-perspective Matching for Natural Language Sentences". In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. IJCAI'17. Melbourne, Aus-

tralia: AAAI Press, pp. 4144–4150. ISBN: 978-0-9992411-0-3. URL: http://dl.acm.org/citation.cfm?id=3171837.3171865.

Weeds, Julie, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller (2014). "Learning to Distinguish Hypernyms and Co-Hyponyms". In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, pp. 2249–2259. URL: https://www.aclweb.org/anthology/C14-1212.

Williams, Adina, Nikita Nangia, and Samuel Bowman (2018). "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 1112–1122. URL: http://aclweb.org/anthology/N18-1101.

Xu, Kelvin, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio (2015). "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention". In: *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, pp. 2048–2057. URL: http://dl.acm.org/citation.cfm?id=3045118.3045336.

Yoon, Deunsol, Dongbok Lee, and SangKeun Lee (2018). "Dynamic Self-Attention : Computing Attention over Words Dynamically for Sentence Embedding". In: *CoRR* abs/1808.07383. arXiv: 1808.07383. URL: http://arxiv.org/abs/1808.07383.

Young, Peter, Alice Lai, Micah Hodosh, and Julia Hockenmaier (2014). "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions". In: *Transactions of the Association for Computational Linguistics* 2, pp. 67–78. ISSN: 2307-387X. URL: https://transacl.org/ojs/index.php/tacl/article/view/229.

Zhang, Zhuosheng, Yuwei Wu, Zuchao Li, Shexia He, Hai Zhao, Xi Zhou, and Xiang Zhou (2018). "I Know What You Want: Semantic Learning for Text Comprehension". In: *arXiv:1809.02794 [cs]*. arXiv: 1809.02794. URL: http://arxiv.org/abs/1809.02794.