University of Amsterdam

MSc Artificial Intelligence

Master Thesis

Textual (Generalised) Any-Shot Learning
The Case of Relation Classification

---

by

Nuno Neto de Carvalho Mota

11413344

August 16, 2019

36 European Credits

November 2018 - August 2019

Supervisor/Examiner:
Dr. Wilker Ferreira Aziz

Co-supervisor:
Dr. Miguel A. Rios Gaona

Assessor:
Dr. Elia Bruni

Institute for Logic, Language and Computation

Acknowledgements

Any work hardly ever consists of the efforts of a single person. This thesis is not in anyway different from the norm. It represents the contributions, however direct or indirect, of several people that deserve due acknowledgement.

First and foremost it represents the combined efforts of my parents, who have not only enabled my studies, and long have they been, but have also supported me along the entire way, offering counsel and guidance, while still fully providing me with the freedom to explore and pursue that which fascinated me. The same can be said of my stepdad, my siblings and my extended family in general, who have been, to a higher or lower degree, ever present.

Second, my supervisor Wilker and my co-supervisor Miguel have taken the brunt of the direct contributions, always with patience and with the knowledge to point me to where I should direct my efforts. Their suggestions have been precious and have effectively made this work possible.

Third, I would like to thank Elia for taking the time to assess my work. I would also like to thank Abiola Obamuyide and Andreas Vlachos for making their data available to us. Likewise, I also thank Zeynep Akata for taking the time to clarify questions I had about Generalised Any-Shot Learning. Equally as important, I would like to thank Haitam for the time he took in reviewing my work and, along with others such as Bram, Victor, Alexandra and Marco, the insightful discussions that helped shape the end result (let us also not forget the cheeky, spirit-lifting foosball games).

Forth, I would like to thank Anouk, for there is no hobbit merrier than her, with just the right amount of matching craziness.

Finally, I have to thank all my friends, those that are close and those that are far, even if only in terms of distance. Those close by, from the MLK 404 goons to those with whom I cross paths not as often as I should, have turned Amsterdam into a fantastic place, that made this entire MSc that much more enjoyable. Those afar have shown that distance means little, even when I got too abstracted in my work and was not as present as I should. Knowing that provided a calm that allowed me to more easily complete my studies.

All in all, each and every contribution has been essential for the completion of this thesis and, for that and more, I thank you.

Abstract

In this thesis we introduce, *to our knowledge*, the tasks of Generalised Any-Shot Learning in both the Relation Classification literature and the General Natural Language Processing literature. This is a more realistic and harder task, as Any-Shot learning is by design artificially easy.

As no standard evaluation splits exist, we create new ones, borrowing design decisions from the Computer Vision literature, which has a more developed understanding of Any-Shot Learning and its corresponding Generalised counterpart. We hope that this newly proposed splits will provide a viable means of comparison for future research.

Additionally, we expand upon existing research by turning a binary classification Natural Language Inference (NLI) based model into one that learns to directly predict a categorical distribution over relations.

Furthermore, we compare our main model with a far simpler one, that is also capable of performing (Generalised) Any-Shot Learning tasks. This way, we were capable of determining that our main model, which is a more complex and proven NLI based architecture, does indeed bring significant benefits to the (Generalised) Any-Shot Learning Tasks. By comparing both models we also establish baselines for future research in our proposed splits.

Moreover, we determined that training models for the tasks of (Generalised) Zero-Shot Learning under an Open Set Framework, as opposed to a Closed Set one, significantly improves performance on the *unseen* classes.

Finally, we also determined that our models are quite capable of correctly identifying relations present in sentences that have no specific annotation.

Table of Contents

List of Figures

List of Tables

Acronyms

AE	Auto-Encoder
AI	Artificial Intelligence
ALE	Aligned Latent Embeddings
ASL	Any-Shot Learning
BiLSTM	Bidirectional LSTM
BoW	Bag of Words
CNN	Convolutional Neural Network
CV	Computer Vision
DAP	Direct Attribute Prediction
DEBUG	Debugging
DL	Deep Learning
ESIM	Enhanced Sequential Inference Model
FE	Final Evaluation
FSL	Few-Shot Learning
GASL	Generalised Any-Shot Learning
(G)ASL	(Generalised) Any-Shot Learning
GFSL	Generalised Few-Shot Learning
(G)FSL	(Generalised) Few-Shot Learning
GZSL	Generalised Zero-Shot Learning
(G)ZSL	(Generalised) Zero-Shot Learning
HT	Hyperparameter Tuning
IAP	Indirect Attribute Prediction
IE	Information Extraction
KB	Knowledge Base
KL	Kullback–Leibler divergence
LeakyReLU	Leaky Rectified Linear Unit
LM	Language Model
LSTM	Long Short Term Memory RNN
MAP	Maximum a Posteriori

ML	Machine Learning
MLP	Multi Layer Perceptron
MMD	Maximum Mean Discrepancy
MultiNLI	Multi-Genre Natural Language Inference
NER	Named Entity Recognition
NL	Normal Learning
NLI	Natural Language Inference
NLP	Natural Language Processing
NLU	Natural Language Understanding
NSE	Neural Semantic Encoder
POS	Part of Speech
QA	Question Answering
RC	Relation Classification
RE	Relation Extraction
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SLU	Spoken Language Understanding
SNLI	Stanford Natural Language Inference
VAE	Variational Auto-Encoder
VI	Variational Inference
ZSL	Zero-Shot Learning

Chapter 1

Introduction

In the last few years we have seen a resurgence of Artificial Intelligence (AI)[1] that has provided society with ever more powerful and, often enough, useful tools. In fact, the breadth of possible applications for AI systems is nearly only limited by our imagination, ranging from self-driving cars, translation systems, assisted farming, personal assistants to even just plain entertainment oriented applications.

[1] *Due, in great part, to Machine Learning (ML).*

Nonetheless, while it seems like AI is here to stay, there's still much to be explored and to be improved. For example, one common pitfall of most AIs is their lack of knowledge of the world, particularly when performing general reasoning. On top of that, they are also usually incapable of extrapolating to unseen scenarios, effectively making them highly specialized in the task they were originally designed for, but nearly useless in other tasks.

One possible way to mitigate the first issue is by providing AIs with some kind of persistent and easily interpretable form of factual world knowledge. This can be achieved by, for example, the use of Knowledge Bases (KBs), which are actually also useful for human users, as KBs can contain vast amounts of information, much more so than what the average person can remember or even learn during their lifetime. However, this might raise the eyebrows of some of the readers, leading them to wonder that "*if KBs are so vast, wouldn't the manual creation of one be an extremely hard, laborious and expensive task?*". The answer is yes. Fortunately, AI can also help with this task!

Information Extraction (IE) aims exactly at tackling this issue, by attempting to automatically extract facts from unstructured data, usually in the form of text[2], and using those facts to populate KBs. Since its appearance, IE has become a vast and active field of scientific research. Delving into its entirety would far exceed the scope of this Master Thesis. As such, the focus is shifted towards the more restricted area of Relation Extraction (RE) and, in particular, Relation Classification (RC). This subfield of IE only aims at modelling factual relationships between entities and while RE is concerned with both *detecting* entities and *classifying* the relations between them, RC is only concerned with the classification aspect of the task[3].

[2] *Which can easily be found on the web, for example.*

[3] *The entities are presumed to be known in advance.*

Previous research (Obamuyide and Vlachos, 2018, [46]) took the RE work, and data, of Levy et al. (2017, [37]) and adapted it in order to make

use of NLI modules for RC. Simply speaking, they use a NLI module to classify whether a sentence, x, entails a relation description, y, thus determining whether that specific relation is observed in x or not. This means that *for each relation* they perform a binary classification[4] between x and the corresponding relation's description, y. With this formalisation of RC it is possible to inaccurately identify the presence of multiple disjoint relations.

[4] *They remove the neutral label commonly present in NLI tasks.*

Contribution 1: **This thesis proposes shifting the NLI module from directly performing the binary classification to instead becoming a learnable scoring function**.

This allows the reinterpretation of the problem as one of categorical classification of the possible relations present in a sentence, making the task of prediction easier and cleaner.

Contribution 2: Additionally, **we determine to what extent complex modules specifically designed for NLI are beneficial to the task of RC, in favour of significantly simpler architectures**.

The works of Levy et al. (2017, [37]) and Obamuyide and Vlachos (2018, [46])[5] also investigate their methods' ability to extrapolate to previously unseen settings. In order to do so, they evaluate their performance under particularly harsh evaluation settings, where at training time some of the classes, identified as *unseen* classes, might have no instances whatsoever (Zero-Shot Learning (ZSL)) or only a minute amount of instances (Few-Shot Learning (FSL)). An important aspect of these evaluation settings is that at test time it is assumed that instances belong exclusively to *unseen* classes.

[5] *Among other works.*

We point out[6] that these are in fact artificially easy, and hardly realistic, tasks, as it essentially allows algorithms to sidestep the data imbalance problem encountered at training time.

[6] *Based on Computer Vision (CV) literature, which, compared to Natural Language Processing (NLP), has devoted more attention to this specific line of research.*

Contribution 3: As such, *to our knowledge*, **this thesis introduces in both the Relation Classification (RC) and the general Natural Language Processing (NLP) literatures, the harder and more realistic task of Generalised Any-Shot Learning (GASL)**.

The Generalised case loses the assumption that at test time any instance belongs exclusively to one of the *unseen* classes. Now the algorithm needs to classify between an instance belonging to any of the classes in $seen \bigcup unseen$, which makes it much harder to correctly classify the *unseen* instances.

Contribution 4: We note that no standard splits exist for most of these evaluation settings. Therefore, and also as a way to evaluate our own method, **we propose a series of evaluation splits for the task of (Generalised) Any-Shot Learning**, making use of the data collected by Levy et al. (2017, [37]). We hope this will render the comparison of future research more realistic.

Finally, we take the first steps, under our framework, into investigating ways of learning how to classify relations[7] in an unsupervised way.

The structure of this thesis is as follows:
Chapter 2 reviews relevant background knowledge and literature. Namely Natural Language Inference (NLI), Relation Extraction (RE)/Relation Classification (RC) and an in depth[8] discussion of (Generalised) Any-Shot Learning ((G)ASL)[9] are presented. In chapter 3 the construction of the proposed GASL evaluation splits is discussed. Afterwards, in chapter 4 our NLI based RC method is discussed in depth, along with the performed evaluation on the newly proposed GASL splits. That is followed by the discussion of conclusions and of potential future work, in chapter 5.

[7] *In fact our method is general enough that it could potentially be applied to a diverse range of text based classification tasks.*

[8] *As, of the three, (Generalised) Any-Shot Learning ((G)ASL) is the research area most foreign to NLP*

[9] *The (G) stands for both the Generalised case and the non-Generalised case.*

Chapter 2

Background

Before jumping directly into the approach put forth in this thesis, it may be beneficial for some of the readers to get better acquainted with certain background topics. As such, three main topics are presented.

In section 2.1 a brief introduction to NLI is presented.

Afterwards, section 2.2 introduces the NLP task of predicting relations, between entities, in text.

To conclude, in 2.3, classification training scenarios where data is extremely sparse or even unavailable are discussed. In concrete, Zero-Shot Learning (ZSL) (2.3.1), Few-Shot Learning (FSL) (2.3.2) and their Generalised counterpart (2.3.3) are examined. As a closing point, 2.3.4 presents an overview of some of the most common methods in this research area.

2.1 Natural Language Inference

Natural Language Inference (NLI) is only a part of the larger, more involved task of Natural Language Understanding (NLU). NLU is concerned with how AI systems interpret and understand the structures and meaning behind language, particularly in a way that allows users[10] to interact with the AI systems in a conversational manner. On the other hand, NLI is concerned specifically with being able to have AI systems understand logical *entailment* inferred from Natural Language propositions. Concretely, NLI aims at being able to determine whether a certain hypothesis h can be entailed from the context implied by a premise p, in such a way that the NLI system can claim that h is a *contradiction* given p, that h is *neutral* with respect to p or that h can be *entailed* from p.

[10] *That is, humans.*

What does this mean exactly? Perhaps the best way to explain it is to illustrate it with specific examples. Consider the following:

- Premise p: *John is standing outside, in the rain, without an umbrella.*
- Hypothesis *h1*: *John's clothes are wet.*
- Hypothesis *h2*: *John likes French food.*
- Hypothesis *h3*: *John's clothes are dry.*

It is fairly easy for any person to understand that if *John is standing outside, in the rain, without an umbrella* then his clothes will most likely be wet, as we have knowledge of the world that helps us in reasoning about the consequences of being in the rain. As such it is easy for anyone to conclude that *h1* can be and is *entailed* from *p*. Likewise, our knowledge of the concepts of *liking* and *French food* allow us to easily understand that *h2* has nothing to do with *p*. Consequently, we understand that the truth of the statement *John likes French food* cannot be verified given *p*, which makes *h2 neutral* with respect to *p*. Finally, with a similar reasoning as the one presented for *h1*, it is expectable that everyone would correctly came to the conclusion that *h3* is most likely a *contradiction* given the premise *p*.

While it might be easy, in general, for humans to ascertain whether a hypothesis can be inferred from a premise or not, it is much harder for an AI system to do so. Any AI system would have to at least understand the meaning behind certain components of both the premise and the hypothesis and also be able to reason about them and the relations between them.

Early works on NLI relied on extremely small datasets (Dagan et al., 2006, [15])[11] and required hand-crafted features devised by experts, along with heavily tailored models. Such models would be based on, *for example*, the translation of Natural Language to First-Order Logic and subsequent application of theorem provers (Akhmatova, 2005, [2]), with potential extra considerations, e.g. including external KBs (Fowler et al., 2005, [19]).

The introduction of the Stanford Natural Language Inference (SNLI) corpus[12] (Bowman et al., 2015, [8]), which is about 2 orders of magnitude bigger than previous corpora[13], allowed, for the first time, the competitive tackling of NLI by Deep Learning (DL) models.

Bowman et al. (2015, [8]) also introduced a very simple baseline with which to test for entailment. It consisted of creating sentence embeddings, of both *p* and *h*, using Long Short Term Memory RNNs (LSTMs) (Hochreiter and Schmidhuber, 1997, [30]), followed by their concatenation, which was then input to a Multi Layer Perceptron (MLP) classifier. While this model's performance was already quite good, more complex models soon appeared.

Rocktäschel et al. (2016, [58]) extended the approach of Bowman et al. (2015, [8]) by initialising the cell state of the LSTM that produces the hypothesis' sentence embedding with the premise's sentence embedding, effectively conditioning the representation of *h* on the given *p*.

More importantly the authors introduced attention (Bahdanau et al., 2015, [4]) in the NLI literature. On a first approach, they extended their previous method by using the hypothesis' sentence embedding and the

[11] *Dagan et al. (2006, [15]) also provide an overview and comparison between several different methods applied to their PASCAL RTE dataset.*

[12] *And its broader coverage extension, the Multi-Genre Natural Language Inference (MultiNLI) corpus (Williams et al., 2018, [71]).*

[13] *With around 570K paired premise-hypothesis examples.*

premise's word-level LSTM's hidden states to produce attention weights over the *premise*. These were then used to produce a new attention-weighed sentence embedding of p[14]. They also implemented a version where each hypothesis' word-level hidden stage created attention weights over the *premise*, thus creating a representation of p for each word of h. These were finally combined[15] into a single sentence embedding of p[14]. Finally, they discussed two-way attention, by employing their attention methods for each direction of Bidirectional LSTMs (BiLSTMs) (Graves and Schmidhuber, 2005, [24]).

[14] *The classification procedure was as before.*

[15] *Through the usage of a Recurrent Neural Network (RNN) model.*

The use of attention is quite interesting. While the simple overall sentence embeddings may represent reasonably well the meaning of a sentence, it is often the case that for NLI only a small number of elements of the premise/hypothesis are necessary to correctly infer whether h is entailed from p or not. Seeing as the attached meaning of these important elements can be easily subsumed by the other elements of the sentence[16], by using attention, models can learn how to match elements that are potentially more relevant for the task at hand, allowing for more adequate sentence embeddings or even more involved mechanisms, as is the case of most works following that of Rocktäschel et al. (2016, [58]).

[16] *When producing a single sentence embedding.*

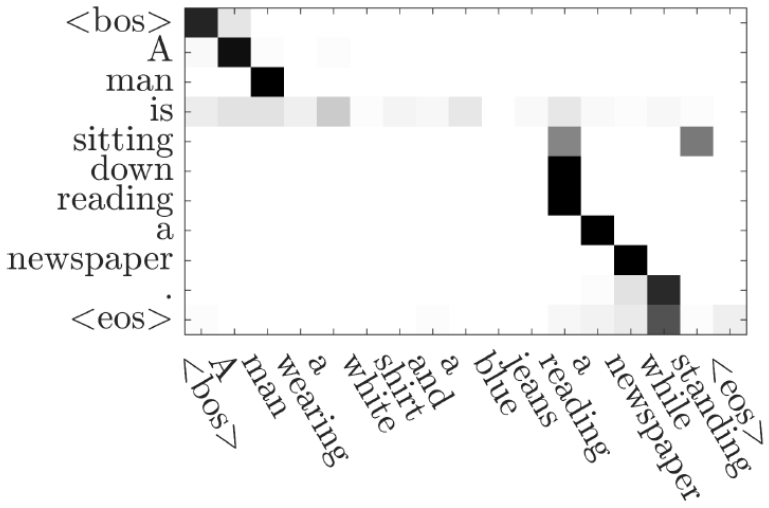


Figure 2.1: Attention

Example of (row) normalised word-by-word attention weights between two sentences, highlighting the importance of certain components when matching the sentences. Adapted from Chen et al. (2017, [11]).

Wang and Jiang (2016, [69]) base themselves on Rocktäschel et al. (2016, [58]), but start by dropping the conditioning of h on p, as they argue that it is preferable to get an embedding for each of them separately and combine them later. They still produce word-level hidden states for both p and h, along with the attention vectors of the second attention method of

Rocktäschel et al. (2016, [58]), but argue that using a single sentence embedding of p to match h is not ideal. As such, they proposed a second *matching* LSTM that takes as input the concatenation of the attention-weighed premise's embeddings with their corresponding hypothesis' word-level hidden embeddings, thus allowing h to also be matched with p at the word level and to have the LSTM learn which matches to give importance to.

Parikh et al. (2016, [48]) propose to instead shift the attention mechanism uniquely to the word level[17]. By making use of pre-trained GloVe word embeddings (Pennington et al., 2014, [50]), they compute word-by-word attention weights between both p and h, which are then used to produce new attention-weighed word embeddings. They combine each original word embedding with it's corresponding attention-weighed version[18] and, afterwards, aggregate them at the sentence level. These final sentence embeddings are concatenated and classified with the usual MLP procedure.

Alternatively, Munkhdalai and Yu (2017, [43]) are able to achieve state of the art results through the usage of their complex, memory based Neural Semantic Encoder (NSE)[19]. Both p and h have an associated NSE, with an additional shared memory between them, allowing the encoding of information given one another. Entailment is estimated as usual[20].

Chen et al. (2017, [11]) expand upon the attention mechanism proposed by Parikh et al. (2016, [48]) and introduce the Enhanced Sequential Inference Model (ESIM) model. They start by producing context-aware input word embeddings, by using different BiLSTMs for p and h, followed by the word-by-word attention mechanism of Parikh et al.. They compute a representation of each sentence's word by concatenating the original context aware word embedding, the attention-weighed one, their difference and their product. They then perform what they denominated as the *inference composition* phase, where they pass the concatenated word-level vectors through two other BiLSTMs[21], and finally produce a sentence-pair representation by concatenating the average of the final word embeddings along with their *maxpooled*[22] version, for both p and h, which is finally given as input to a classifier MLP. They also present a version that makes use of syntactic parse trees, encoded with tree-LSTMs, which marginally improves their previous results. The ESIM model will be further discussed on following sections, as it is the core component of the proposed approach, even though we abstain from using its syntactic version.

There has been quite some work researched in NLI after that of Chen et al. (2017, [11]). These include, *for example*, the work of Radford et al. (2018, [54]), who rely on pre-training a transformer (Vaswani et al., 2017, [67]) as a high-capacity Language Model (LM) and then fine-tuning it for different independent tasks, including NLI. Also, several NLI techniques

[17] *Allowing them to reduce the number of trainable parameters by an order of magnitude.*

[18] *They are combined by employing a MLP on their concatenation.*

[19] *Which produces a sentence embedding.*

[20] *Using the concatenation/MLP method*

[21] *One for the premise and another for the hypothesis.*

[22] *At each embedding dimension.*

focus on sentence embedding learning models (Nie and Bansal, 2017, [44]) (Balazs et al., 2017, [5]) (Chen et al., 2017, [12]) (Wang et al., 2017, [70]), yielding better representations with which to perform NLI. Interestingly, Conneau et al. (2017, [13]) show that NLI based sentence embeddings outperform several regular embedding techniques.

However, ESIM has been shown to be relatively competitive to even newer methods, while often requiring orders of magnitude fewer parameters. Additionally, the main related work that this thesis is based on (Obamuyide and Vlachos, 2018, [46]) also makes use of ESIM. While more advanced NLI architectures could be investigated in this thesis, that is beyond the scope of this work and such research is left for future work.

Now that a clear overview of NLI has been provided, the tasks of Relation Extraction (RE) and Relation Classification (RC) will be presented.

2.2 Relation Extraction/Classification

The field of Relation Extraction (RE) is concerned with extracting factual relationships, between entities, from unstructured text. While we have mentioned factual relationships before, some readers might not have a clear idea of what this actually means. As such, consider the following example:

"Barack Obama, the 44th president of the United States of America, is married to Michelle Obama."

This example contains three entities, namely **Barack Obama**, **United States of America** (**U.S.A.**) and **Michelle Obama**. Also, exactly two factual relationships exist, which can be represented by the relational tuples[23] **(Barack_Obama, president_of, U.S.A.)** and **(Barack_Obama, married_to, Michelle_Obama)**. On the other hand, it also contains non-factual, or false, relationships, as, for example, **(Michelle_Obama, president_of, U.S.A.)**. Note that we restrict ourselves to binary relationships[24], but that does not need to be the case.

Now that the concept of a factual relationship has hopefully been made more clear, Relation Extraction (RE) can be properly described. Given an unstructured text source, S, RE usually consists of two stages:

- First, any possibly relevant entities $e \in \mathcal{E} \wedge e \in S$ are identified[25], using, for example, a Named Entity Recognition (NER) algorithm.
- Second, the relationship, if any, between each pair of identified entities $(e_i, e_j)_{i \neq j}$ is determined.

[23] *A relational tuple (entity_1, relation, entity_2) can also be represented in a way more similar to that of KBs:* `relation(entity_1, entity_2)`.

[24] *A binary relation is a relation between* two, *and specifically* two, *entities.*

[25] *Where $\mathcal{E}$ represents the set of all possible entities.*

Reusing the previous example, a RE system would have to identify all three entities and then would need to come up with only the two valid factual relationships. Hopefully, it would avoid getting any false positive hits, such as the previously mentioned **(Michelle_Obama, president_of, U.S.A.)**. In fact any system would not only need to identify that a factual relationship exists, but also correctly determine what that relationship is, avoiding, for example, **(Barack_Obama, film_editor, U.S.A.)**.

There have been several proposed approaches to this problem. Banko et al. (2007, [6]) introduce OpenIE, where they train a Naive Bayes Classifier on Part of Speech (POS) based features, in order to determine whether a pair of entities and the text between them represent a valid relational tuple. At test time[26] a POS tagger is run and a noun phrase chunker identifies relevant entities. The POS features of the entities and the text between them are then used[27] to determine the validity of the proposed relational tuple. This method assumes that the surface pattern of whatever lies between the entities is a relation mention, essentialy making it a *schemaless*[28] approach.

[26] *Or extraction time.*

[27] *By the Naive Bayes Classifier.*

[28] *A schema represents the set of relations.*

Etzioni et al. (2011, [18]) note that such an approach can lead to incoherent or uninformative extractions, due to misidentification of the constituent words of both the relation's surface form and the entities. They address these issues by focusing on verb based relation phrases and by training a better entity identifier.

Riedel et al. (2013, [56]) combine the surface form relations of OpenIE with structured knowledge present in already existing KBs, such as Freebase (Bollacker et al., 2008, [7]), yielding what they call a *Universal Schema* approach. They model the problem as a matrix, where columns represent relations and rows represent entity pairs. By associating both rows and columns, individually, with a specific feature vector, they can learn a classifier for determining whether each cell in the matrix represents a valid relational tuple or not. Their approach also allows them to model *implicature*[29], since each entity pair can be associated with several relations.

[29] *For example,* `X professor-at Y` *implies* `X employer-at Y.`

One downside of these methods is that they can lead to an extremely large number of identified relations, many of them semantically equivalent, but with different surface patterns. It can thus be desirable to extract instead a canonical value that represents a relation, independently of how that relation is expressed in text. One way to address this issue is by determining relations, from a predefined set of admissible relations. The method of Riedel et al. (2013, [56]) could potentially address this issue, if only KB relations were to be used[30].

[30] *And semantically equivalent relations, present in distinct KBs, were to be reduced to one canonical relation type.*

Alternatively, Levy et al. (2017, [37]) pose the entire problem as that of Question Answering (QA). For each canonical relation type they create

a set of questions that admit a subject entity. Their method consists of determining for which questions[31] a non-null answer can be found. For those, the relation corresponding to the question is extracted, while if no answer is found, they assume that the corresponding relation is not valid.

Closer to the work of this thesis is that of Obamuyide and Vlachos (2018, [46]), who turn the relation related questions of Levy et al. (2017, [37]) into statements concerning arbitrary subject and object entities. This essentially grounds each relation type with a set of corresponding descriptions. Having the descriptions allows them to frame the whole task as one of NLI. That is, given a sentence S and a relation description y, they make use of existing NLI methods to test whether S entails y, which is basically checking whether the relation associated to description y can be inferred from S or not.

However, it is possible to determine multiple associated relations and while this allows to implicitly model implicature in the process, this can often lead to a noisy classification process, with multiple wrong relations being identified at test/extraction time.

A more direct and interpretable approach is to turn the problem into a categorical classification paradigm[32], which is what we propose. In this case, Relation Classification (RC) can simply be defined as:

- Given a sentence S, with a pair of annotated entities, e_1 and e_2, the task is to determine, from a set of predefined relations $\mathcal{R}$, the semantic relation r that best relates e_1 and e_2.

While relying on a set of predefined relations can avoid extracting semantically equivalent surface forms, the truth is that designing systems in such a way has it downsides. In particular, systems like that of Banko et al. (2007, [6]), Etzioni et al. (2011, [18]) and Riedel et al. (2013, [56]) are not restricted to predicting previously *seen* relations[33], giving them the flexibility to adapt to new contexts[34]. It would be ideal if it is was possible to somehow avoid the semantic equivalence problem and still be able to extrapolate to previously *unseen* concepts. Unlike most methods that predict relations out of a predefined set, the works of Levy et al. (2017, [37]) and Obamuyide and Vlachos (2018, [46]) can actually also predict *unseen* relations. However, the authors only test their models' performance under Any-Shot Learning (ASL) settings, which does not consist of the most realistic approach. Let us now present, in some depth, (Generalised) Any-Shot Learning ((G)ASL), and the reason of why ASL is not realistic will soon become apparent.

[31] *And, consequently, the associated relations.*

[32] *In this case implicature can, somehow, be determined between the relations' canonical values, instead of being directly inferred from data.*

[33] *By extracting different surface patterns or adding a column representing a new relation.*

[34] *While they can adapt to new contexts, it has been shown they they do not generalise very well.*

2.3 (Generalised) Any-Shot Learning

While AI, and in specific ML, algorithms perform increasingly more impressively as time passes, they still falter when attempting to mimic our inherent ability to recognise previously unobserved (or rarely observed) objects/concepts for which we have at least been given a description.

Perhaps the main reason we are so good at this is because we already have a "model" of some concepts, from which we can extrapolate information in order to differentiate and distinguish new unseen/rare ones. This allows us to mainly focus on the expected distinguishing features of the new concepts and to use the implicit probability distribution of the known concepts' features as a strong surrogate to the new ones (Miller, 2002, [41]).

Fundamentally improving our AI systems to better resemble the way humans learn is undoubtedly an interesting motivation, but there is also a practical one that drives the investigation of Any-Shot Learning (ASL): the data collection and data labelling problems. When it comes to classification tasks[35] it is not uncommon for labelled data to be naturally scarce for some classes. This is even more accentuated the more classes considered, as it becomes more impractical[36] to get data on some of them or/and to have expert annotators who accurately annotate all classes. As such, ASL methods define a separation: common classes are denominated as *seen* classes, $\mathcal{Y}^S$, and rare ones are denominated as *unseen* classes, $\mathcal{Y}^U$, such that[37]:

$$\mathcal{Y}^S \subset \mathcal{Y} \ \wedge \ \mathcal{Y}^U \subset \mathcal{Y} \wedge \ \mathcal{Y}^S \cap \mathcal{Y}^U = \emptyset \tag{2.1}$$

Consider, for example, the image classification task, where it is easy to get pictures of cars, but quite difficult to do so for nearly extinct animals.

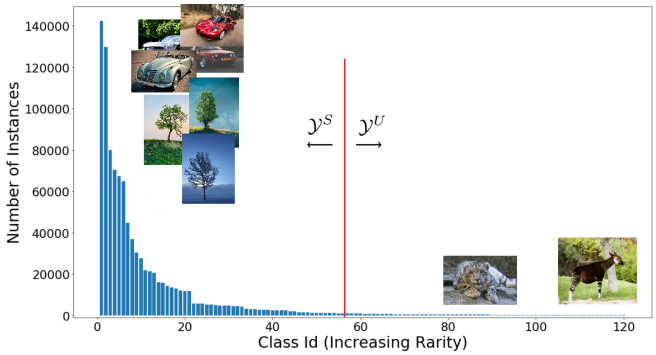


Figure 2.2: Long Tail Data Imbalance

Illustration of the data frequency imbalance between different classes.

These motivations led to the beginning of research on Zero-Shot Learning (Larochelle et al., 2008, [36]) and Few-Shot Learning[38] (Miller et al., 2000, [40]), fields of research that have bloomed in the recent years and which shall now be reviewed, along with their generalised counterpart.

[35] *Which are the tasks that ASL methods tackle.*

[36] *And probably prohibitively expensive.*

[37] *With $\mathcal{Y}$ being the set of all possibly relevant classes.*

[38] *In fact Miller et al. (2000, [40]) only researched One-Shot Learning.*

2.3.1 Zero-Shot Learning

Zero-Shot Learning (ZSL) is meant to be the hardest of the Any-Shot Learning (ASL) settings. This is the setting where, at training time, models are presented with no instances of the *unseen* test classes[39]. This raises the question: if the algorithms are shown no instances of the *unseen* classes, how can they possibly transfer knowledge from the *seen* to the *unseen* classes?

Currently, the solution is to associate each class with some kind of side information that describes it. This way models can learn how to relate the data space to the class space. At test time, methods are, hopefully, capable of interpreting the *unseen* classes' side information in such a way that they can relate it to the corresponding data instances of those classes.

[39] *Transductive ZSL allows the presence of unlabelled instances of the* unseen *classes at training time. However, we focus on Inductive ZSL, where no instances of the* unseen *classes, whether labelled or not, are present at training time.*

Formally, given a training dataset $\mathcal{D} = \left\{\left(x_n, y_n \in \mathcal{Y}^S\right)\right\}_{n=1}^{N}$, where x_n is the n^{th} instance of the dataset, y_n is its corresponding class identifier, $\mathcal{Y}^S$ is the set of *seen* classes[40] and N is the total number of instances in the dataset, the task of the model is to learn a mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X}$ is the data space and $\mathcal{Y}$ is the set of all relevant classes, *seen* and *unseen*. The mapping f is defined as (Xian et al., 2017, [73]):

[40] *The classes that have labelled instances at training time.*

$$f\left(x; \boldsymbol{W}\right) = \underset{y' \in \mathcal{Y}}{\arg\max}\ F\left(x, y'; \boldsymbol{W}\right) \tag{2.2}$$

where F is a score function[41] and $\boldsymbol{W}$ represents the model's parameters.

[41] *Here it is implicitly assumed that given a specific class y, F can directly access the corresponding side information.*

Interestingly, ZSL allows framing the training procedure in two ways: one where all **classes that are known to exist**[42] correspond only to a **subset of all possible classes**, i.e. an open set framework; and another where all **classes that are known to exist**[42] correspond in fact the **full set of all possible classes**, i.e. a closed set framework.

[42] *Both seen (train) and unseen (test) classes.*

Open Set Framework

Scheirer et al. (2012, [59]) argue that for some problems we do not need, and most times cannot have, knowledge of the entire set of possible classes. The authors claim that from the perspective of applications the open set framework is a more realistic scenario, where it is often the case that at training time there is incomplete knowledge of the world. Thus, it would be desirable to at test time be able to submit previously unknown classes to the AI system, without having to retrain it.

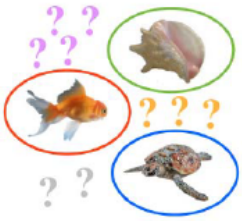


Figure 2.3: Open Set Framework

Adapted from Scheirer et al. (2012, [59]).

This open set framework goes more in line with the first motivation presented for the study of ASL, where the point is for AI systems to better resemble human intelligence, allowing them to learn about new concepts on the fly. Formally:

$$\mathcal{Y}^S \cup \mathcal{Y}^U \subset \mathcal{Y}$$

From a mathematical perspective, this means that the training objective to be minimised can be defined as[43]:

$$\frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}} L\left(y,\ \underset{y'\in\mathcal{Y}^S}{\arg\max}\ F\left(x,y';\boldsymbol{W}\right)\right) \tag{2.3}$$

where $L\left(\cdot\right)$ is the loss function.

Framing classification problems in an Open Set Framework is often referred to as Recognition, instead of classification.

[43] *Note that the argmax is only taken over* $\mathcal{Y}^S$.

Closed Set Framework

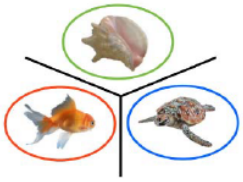


Figure 2.4: Closed Set Framework

Adapted from Scheirer et al. (2012, [59]).

On the other hand, when posing the problem as a Closed Set Framework it is assumed that the set of all classes that are possibly relevant is known a priori. This is the case in which classification problems are usually framed. Thinking back again to the initial ASL motivations, presented at the beginning of this section, one can see that framing ZSL problems under the Closed Set assumption goes more in line with the second motivation, where the problem lies in the difficulty with collecting/annotating data for all classes. Formally:

$$\mathcal{Y}^S \cup \mathcal{Y}^U = \mathcal{Y}$$

From a mathematical perspective, this means that the training objective to be minimised can be defined as[44]:

$$\frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}} L\left(y,\ \underset{y'\in\mathcal{Y}^S\cup\mathcal{Y}^U}{\arg\max}\ F\left(x,y';\boldsymbol{W}\right)\right) \tag{2.4}$$

where $L\left(\cdot\right)$ is the loss function.

[44] *Note that the argmax is taken not only over* $\mathcal{Y}^S$ *but also* $\mathcal{Y}^U$*, even though no instances of* $\mathcal{Y}^U$ *will exist at training time.*

The distinction between the Open Set Framework and the Closed Set Framework is an important one to make, as most classification algorithms have a very strong bias towards the more populated classes. Seeing as most examples belong to *seen* classes[45], what happens in the Closed Set Framework is that the model consistently gets a negative learning signal for the *unseen* classes, effectively learning to almost never predict said classes. On the other hand, this does not happen in the Open Set Framework. This is something that is verified later on, in the performed experiments (4.3.3).

[45] *For Zero-Shot Learning that is actually all examples.*

Finally, while the training of ZSL methods can follow different approaches, the testing is simple and straightforward: ZSL methods are simply concerned with classifying instances that are presumed to belong **exclusively** to *unseen* classes. That is:

$$y* = \underset{y'\in\mathcal{Y}^U}{\arg\max}\ F\left(x*,y';\boldsymbol{W}\right) \tag{2.5}$$

2.3.2 Few-Shot Learning

Few-Shot Learning (FSL) is extremely similar to ZSL, as one might imagine. The main difference is that in FSL the *unseen* classes have a certain number[46] of training instances present at training time[47]. It is important to notice that due to the presence of *unseen* classes' training instances, FSL methods do not necessarily require any sort of side information, as ZSL does. However, this side information is still helpful. Seeing as the work presented in this thesis makes use of such side information, the focus shall be exclusively on that variant of FSL.

[46] *Essentially the Shot number.*
1-Shot - 1 instance
2-Shot - 2 instances
etc.
Common Shot values range from 1 to at most, usually, 10.

[47] *Effectively no longer rendering the* unseen *classes* unseen*, but the naming convention is useful nonetheless.*

Any-Shot Learning can be also referred to as M-Shot, K-Way, where M stands for the Shot number and K to the number of *unseen* classes. We make use of this notation in order to formally describe the FSL task.

As with ZSL, one would have a training dataset with labelled data from the *seen* classes: $\mathcal{D}_S = \left\{\left(x_n, y_n \in \mathcal{Y}^S\right)\right\}_{n=1}^{N}$.
However, now one would also have labelled data from the *unseen* classes[48]: $\mathcal{D}_U = \bigcup_{k=1}^{K} \left\{\left(x_{km}, y_{km} = k \in \mathcal{Y}^U\right)\right\}_{m=1}^{M}$, such that the entire training dataset would be $\mathcal{D} = \mathcal{D}_S \bigcup \mathcal{D}_U$.

[48] *Where for simplicity we assume that k uniquely identifies a class of the* unseen *class set* $\mathcal{Y}^U$.

As the training set now contains labelled instances of the *unseen* classes, the training objective is necessarily the Closed Set Framework minimisation objective (2.4)[49]:

[49] *Note that the argmax is taken over* $\mathcal{Y}^S$ *and* $\mathcal{Y}^U$.

$$\frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} L\left(y, \underset{y'\in\mathcal{Y}^S \bigcup \mathcal{Y}^U}{\arg\max} F\left(x, y'; \boldsymbol{W}\right)\right)$$

where $L\left(\cdot\right)$ is the loss function.

At test time the objective is still the same as the ZSL one:

$$y* = \underset{y'\in\mathcal{Y}^U}{\arg\max} F\left(x*, y'; \boldsymbol{W}\right)$$

One of the main reasons to in the previous subsection have discussed the Open Set vs Closed Set Frameworks is that FSL methods are always framed in a Closed Set Framework. Consequently, any fair analysis on the impact of going from a ZSL setting to a FSL one, i.e. increasing the number of available training instances of the *unseen* classes, should compare a Closed Set Framework ZSL setting against FSL settings.

Note that while one could use a FSL trained model to also perform Recognition, the truth is that the way FSL methods are trained and evaluated, implies that the performance of the method is always assessed under a Closed Set Framework. This is due to the test/*unseen* classes effectively being present at training time.

2.3.3 The Generalised Case

In more recent years plain ASL methods have been criticised (Chao et al., 2016, [10]) (Xian et al., 2017, [73]) as being a restrictive set up, where the tasks are artificially easy, and hardly realistic.

This is unsurprising, as in a real application scenario it would be highly unlikely that it would be known a priori whether the encountered data instances would belong exclusively to the *unseen* classes or not.

It is also easy to see why the tasks are artificially easy. Consider a model that performs random classification at test time[50]: in an ASL setting the model is bound to, on average, get at least some of the instances correctly labelled. On the other hand, in a real scenario, the model would have to also discern between *seen* classes, which will have had the most labelled instances at training time, thus probably biasing the model towards classifying any instance as being more likely to belong to one of the *seen* classes.

What one realises is that the plain ASL settings allow models to completely sidestep the data imbalance problem[51], when evaluating the models' performance on the *unseen* classes.

In order to make the task more realistic Chao et al. (2016, [10]) proposed Generalised Zero-Shot Learning (GZSL), which naturally extends to Generalised Few-Shot Learning (GFSL)[52]. Their idea is simple: in order to properly evaluate the true performance of an ASL method it is required to relax the constraint that at test time data instances belong exclusively to the *unseen* classes. This can be formalised as the test objective becoming:

$$y* = \underset{y' \in \mathcal{Y}^S \bigcup \mathcal{Y}^U}{\arg\max} F\left(x*, y'; \boldsymbol{W}\right) \tag{2.6}$$

Now, test time data instances, $x*$, can belong to either *seen* or *unseen* classes, thus getting a better sense of the model's performance on both the *seen* and the *unseen* classes, as opposed to only the *unseen* ones.

[50] *That is, only between the* unseen *classes.*

[51] *Observed at training time between the* seen *and the* unseen *classes.*

[52] *Henceforth, when using the acronym for (Generalised) Zero-Shot Learning ((G)ZSL) or the acronym for (Generalised) Few-Shot Learning ((G)FSL) we refer to both the non-generalised and generalised cases. E.g.: (G)ZSL refers to both ZSL and GZSL.*

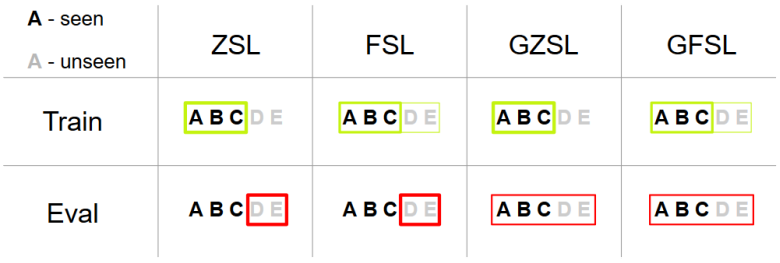


Figure 2.5: Relevant (Generalised) Any-Shot Learning Classes

Example of which classes are considered by each of the different (G)ASL Settings, for both the training phase (Train) and the evaluation phase (Eval).

2.3.4 Most Common Approaches

We shall now discuss some of the most common approaches used for (Generalised) Any-Shot Learning ((G)ASL). These will belong mostly to the CV literature[53], as the bulk of the research that exists has been done there. Some of the few existing NLP works are also discussed.

[53] *We use the work of Xian et al. (2017, [73]) as a guideline, since they themselves performed a relatively extensive literature review.*

The use of side information for scarce data scenarios began with Larochelle et al. (2008, [36]). They introduced the idea of pairing each class with a description vector and learning a mapping from the data space to the description space. This allowed them to simply create description vectors for *unseen* classes, which could then be classified.

Palatucci et al. (2009, [47]) follow the same idea, but use the semantic codes of classes present in a semantic Knowledge Base (KB). For new inputs[54] they predict a semantic code and then determine its corresponding class by finding the KB class that has the best matching semantic code.

[54] *Which can be from a previously* unseen *class.*

Based on a similar idea Lampert et al. (2013, [35]) create a set of binary attributes and pair each class with a unique attribute configuration[55]. With Direct Attribute Prediction (DAP) they learn independent probabilistic classifiers for each attribute, that predict the presence of that attribute in a given data instance. At test time, a new instance is classified by taking a Maximum a Posteriori (MAP) prediction of which class has the most likely attribute configuration, such that:

[55] *Sometimes it can actually be at instance level. If not present for a specific instance, that instance will be paired with the general corresponding class attribute configuration.*

$$F(x, y; \boldsymbol{W}) = \frac{\prod_{m=1}^{M} \mathrm{P}(a_m^y | x, \boldsymbol{W})}{\mathrm{P}(a_m)} \tag{2.7}$$

where a_m^y is the value of the m^{th} attribute of class y and the attribute priors, $\mathrm{P}(a_m)$, are set to the empirical mean observed at training time[56].

[56] *Over all* seen *classes.*

Lampert et al. (2013, [35]) also introduce Indirect Attribute Prediction (IAP). With IAP one first estimates the training class probabilities $\mathrm{P}(Y|x)$. Also, a class attribute matrix $\mathrm{P}(a_m|y_k) = \delta_{a_m^{y_k}, a_m}$, where $\delta_{i,j}$ is the Kronecker delta[57], is defined. At test time they still use 2.7, except that they instead use a newly defined $\mathrm{P}(a_m|x) = \sum_{k=1}^{K} \mathrm{P}(a_m|y_k)\,\mathrm{P}(y_k|x)$, where y_k identifies one of the K training/*seen* classes.

[57] $\delta_{i,j} = 1$ *if* $i = j$
$\delta_{i,j} = 0$ *if* $i \neq j$

Useful as attributes can be, they require defining a new configuration for each new *unseen* class. An easier way, is to make use of rich language embeddings, such as skip-gram embeddings (Mikolov et al., 2013, [39]), to better describe each class.

A later attribute based work is that of Al-Halah et al. (2016, [3]), who attempt to avoid the need to specify a configuration for *unseen* classes. For that, they learn how to match each class' name's semantic embedding with each of its attribute's name's semantic embedding. This allows them to simply provide an *unseen* class' name and automatically infer its attributes.

Completely foregoing attributes, **ConSE** (Norouzi et al., 2013, [45]) trains a plain classifier[58] on the *seen* classes. At test time they combine the semantic embeddings of the *seen* classes, weighed by their respective classification probability, producing a predictive embedding. Using the K-Nearest Neighbours algorithm (Cover and Hart, 1967, [14]), they find the test class that has the semantic embedding closest to the predicted one[59].

On another approach, **DeViSE** (Frome et al., 2013, [20]) retrains the lower layers of a pre-trained N-way classification Convolutional Neural Network (CNN) to instead regress the corresponding classes' semantic embeddings. For prediction they employ a hinge rank-based loss based on a bilinear compatibility function between the predicted semantic embedding and the classes' true semantic embeddings.

Socher et al. (2013, [64]) also use a non-linear projection from the data space to the semantic space. More interestingly, they add a mechanism for novelty detection[60], that allows them to at test time first predict whether an image belongs to the *seen* or *unseen* classes, effectively testing their approach under the Generalised case.

On a different take, Zhang and Saligrama (2015, [76]) learn how to represent both data instances and labels as a proportion of the *seen* classes. That is, they essentially learn how to project both data instances and labels into a common latent space[61] where they try to match these estimates through an Euclidean inner product. At test time they just pick the class whose latent projection best matches the test instance's latent projection.

One NLP work is that of Yazdani and Henderson (2015, [74]), who tackle Spoken Language Understanding (SLU). The objective is to identify actions present in an utterance. For example, *"I am looking for a Chinese restaurant near the centre."* implies the actions `inform(near = centre)`[62] and `inform(food = Chinese)`. Given an utterance and the set of possible action tuples, they compose the utterance and each action tuple into separate semantic embeddings. For each action tuple, its embedding and the utterance's one are used in an inner-product based binary classification, for assertion of the presence of that respective action in the utterance[63].

Changpinyo et al. (2016, [9]) try to directly match the classes' semantic space with that of the models' parameters. They introduce a series of phantom classes[64] and corresponding classifiers, in an attempt to have them generalise equally well to *seen* and *unseen* classes.

With Aligned Latent Embeddings (ALE) Akata et al. (2016, [1]) project both the instances and the classes into separate latent spaces, which they then align through a bilinear compatibility mapping:

$$F(x, y; \boldsymbol{W}) = \boldsymbol{\theta}(x)^T \boldsymbol{W} \boldsymbol{\phi}(y) \tag{2.8}$$

where $\boldsymbol{\theta}(\cdot)$ and $\boldsymbol{\phi}(\cdot)$ are the data and class embeddings, respectively.

[58] *That is, without actually making use of the semantic embeddings.*

[59] *This allows them to evaluate on the generalised case, however, they do not do so.*

[60] *Either through thresholds or outlier detection.*

[61] *Basically, they produce cross-modal embeddings.*

[62] *Where each unique combination* `action(attribute = value)` *is seen as a distinct class. The problem is inherently a many-out-of-many multi-class classification, which diverges slightly from standard ASL settings.*

[63] *Unfortunately the authors only report aggregated results, without distinguishing between the* seen *and* unseen *classes.*

[64] *Not unlike Gaussian Processes'* inducing points *methods (Quiñonero-Candela and Rasmussen, 2005, [52])(Snelson and Ghahramani, 2006, [62]).*

This allows them to avoid optimising an intermediate objective, the attribute based classifiers commonly used in methods like DAP, which they claim is sub-optimal. Using this method they can also correlate the different attributes, unlike DAP, or even simply use other side information sources[65].

[65] *Like semantic embeddings.*

ALE is turned into a non-linear version by Xian et al. (2016, [72]), who introduce a series K-dimensional latent piece-wise bilinear compatibility functions, effectively replacing (2.8) by:

$$F(x, y; \boldsymbol{W}) = \max_{1 \leq i \leq K} \boldsymbol{\theta}(x)^T \boldsymbol{W}_i \boldsymbol{\phi}(y) \tag{2.9}$$

where K is a hyperparameter to be tuned. This way they hope that each of the latent $\boldsymbol{W}_i$ will encode different characteristics of the data.

One disadvantage of CV methods, in the side information approach to (G)ASL settings, is that they need to match data from different modalities, namely image data and textual data. On the other hand, NLP methods are matching textual data with textual data. This means that while NLP methods can use approaches like ALE, and its non-linear version, they can also leverage more specific, specialized and proven architectures, that have been tailored specifically for NLP tasks.

For example, Levy et al. (2017, [37]) use **BiDAF** (Seo et al., 2016, [61]), a model that uses a RNN to encode contextual information for both a sentence and a question, followed by an attention mechanism to align parts of the sentence with parts of the question. The model then predicts probabilities for estimating the start and end of spans, which hopefully represent the answer to the question, if an answer exists. By associating each class with a specific question[66], they can easily define new questions for previously *unseen* classes, allowing them to tackle (G)ASL settings. However, Levy et al. (2017, [37]) only focus on Zero-Shot Learning (ZSL).

[66] *Or even multiple questions, as they actually do.*

Similarly, Obamuyide and Vlachos (2018, [46]) leverage NLI, and in particular the previously discussed ESIM, an architecture that has been shown to yield impressive results for the NLI task. Like Levy et al. (2017, [37]) they can simply devise new class descriptions for previously *unseen* classes, but they only evaluate their method on ASL settings, not GASL settings.

Surprisingly, Relation Classification (RC) is actually the field of NLP that has the most research in ASL[67]. This is in large part due to the introduction of ZSL splits by Levy et al. (2017, [37]) and the more recent FewRel dataset (Han et al., 2018, [26]), which introduced FSL splits. Consequently, recent works have been introduced. For example, Soares et al. (2019, [63]) introduce a BERT embeddings (Devlin et al., 2018, [17]) transformer (Vaswani et al., 2017, [67]) combination, that learns how to match sentences in an unsupervised fashion, by assuming that the same pair of

[67] *To our knowledge.*

entities implies the same relation.

While these works represent a step in the right direction for NLP research in scarce data scenarios, the truth is that, currently, such works can only be evaluated on Any-Shot Learning settings and not on the more realistic Generalised case. We shall now address this issue, by proposing splits that allow for the more realistic task of Generalised Any-Shot Learning.

Chapter 3

Dataset

In this chapter we present our proposed RC splits for both Any-Shot Learning (ASL) and Generalised Any-Shot Learning (GASL). We leverage the public data of Levy et al. (2017, [37]), as it is already suited for the tasks of RE/RC and can be easily tailored for (G)ASL evaluation splits.

In section 3.1 we will quickly describe existing data, followed by, in 3.1.1, the data collection procedure employed by Levy et al. (2017, [37])[68]. We also describe, in 3.1.2, the relations' descriptions leveraged by our NLI method.

Afterwards, in section 3.2 we will fully describe the steps taken in the creation of our proposed splits.

[68] *For a full, in-depth description please consult their work.*

3.1 Existing Data

Several relation related datasets already exist, but these usually have relatively few instances and few classes, making it hard to design proper (G)ASL evaluation splits. These include, for example, the SemEval-2010 Task 8 dataset (Hendrickx et al., 2009, [27]), NYT-10 (Riedel et al., 2010, [55]) and TACRED (Zhang et al., 2017, [75]). Even the newer FSL FewRel dataset (Han et al., 2018, [26]) is itself not huge for Deep Learning based approaches. Furthermore, while oriented for ASL tasks, it does not cover either ZSL nor the Generalised case. Similarly, the dataset gathered by Levy et al. (2017, [37]), **UW-RE**, does not cover FSL or the Generalised case. However, **UW-RE** has more classes and orders of magnitude more instances, making it particularly suitable for Deep Learning approaches. As such, we leverage it for the creation of the (G)ASL evaluation splits. Table 3.1 provides a comparison between the different datasets.

Dataset	Number of Classes	Number of Instances
SemEval	9	6.674
TACRED	42	21.784
NYT-10	57	143.391
FewRel	100	70.000
UW-RE	120	2.4162.56

Table 3.1: Existing Datasets Basic Statistics

Reported results, excluding instances related to no relation (which appear in the original datasets as negative instances). For the first 4 datasets we make use of the statistics reported by Han et al. (2018, [26]).

3.1.1 UW-RE

In order to gather their dataset, Levy et al. (2017, [37]) employ the Distant Supervision paradigm (Mintz et al., 2009, [42]). The Distant Supervision paradigm consists of aligning existing KB resources with unstructured text[69]. Concretely, the intuition behind Distant Supervision is that any sentence that contains two entities that participate in a known KB relation, is likely to express that same relation.

Levy et al. (2017, [37]) use the WikiReading dataset (Hewlett et al., 2016, [28]) to collect their sentences. WikiReading itself aligns each Wikidata[70] (Vrandečić, 2012, [68]) relation **relation(entity_1, entity_2)** with the corresponding **entity_1** Wikipedia article, **D**. Using the Distant Supervision approach, Levy et al. (2017, [37]) consider the first sentence in **D** that expresses both **entity_1** and **entity_2** to be a relation mention of **relation(entity_1, entity_2)**. This process yielded 178 relations with at least 100 distinct mentions.

Since Levy et al. (2017, [37]) are concerned with a Question Answering (QA) formalisation of their RE task, they also create a set of queries[71] for each relation, by employing crowdsourcing. After a query verification phase, they were left with 1.192 high-quality queries, that represent a total of 120 relations[72], for a total of 2.416.256 distinct relation mentions on 1.499.932 unique sentences.

3.1.2 Turning Queries into Relation Descriptions

In this subsection we wish to briefly describe the relations' descriptions that will be used by our NLI module, even though they are not related to the (G)ASL proposed splits.

Like it was previously mentioned, Obamuyide and Vlachos (2018, [46]) make use of NLI in their method, which they test on the data collected by Levy et al. (2017, [37]).

However, since they do NLI instead of QA, they transformed the queries of Levy et al. (2017, [37]) into acceptable NLI *hypothesis*. They did so manually, by converting each query into a statement independent of specific entities, such as "*SUBJECT_ENTITY created OBJECT_ENTITY*", getting a final 1.098 unique descriptions and 1.135 unique relation-description pairs.

We highlight the relation-description **pairs** aspect, as some descriptions actually describe more than one relation[73]. This might seem odd at first. How can we then identify a unique class, in these cases? We will address this problem in chapter 4, but for now we can reason about it from a linguistic ambiguity perspective, as some descriptions can indeed express multiple

[69] *It is worth mentioning that Distant Supervision has been shown to be a somewhat noisy annotation method (Riedel et al., 2010, [55]), particularly when aligning KB to domains different from those that were used to build the KB in the first place.*

[70] *Wikidata is a free collaborative KB.*

[71] *That is, questions.*

[72] *54 relations had no questions that were up to the verification standards.*

[73] *These multi-relation descriptions can be found in Appendix A, table A.1.*

relations. Consider the following two examples:

- *Volvo researched and developed the three-point seatbelt system.*
- *J.R.R. Tolkien's Middle Earth was brought to life through his books.*

If we also consider the relation description *SUBJECT_ENTITY created OBJECT_ENTITY*, it is clear that it can indeed express both of the aforementioned sentences, even though the first sentence can also express relations *researched* and *developed* and the second sentence can express the relations *wrote* and *invented*. This can be seen as a consequence of the expressiveness of language, which can often lead to ambiguities. The attentive reader will have also noticed that this can, in some cases, be interpreted as a case of the previously discussed *implicature*. However, as was also mentioned, in this work we do not address the study of this property.

One consideration to have, regarding the relations' descriptions of Obamuyide and Vlachos (2018, [46]), is that the descriptions are not evenly distributed amongst the relations. In fact, some relations have significantly more descriptions than others, leading to a Long Tail distribution:

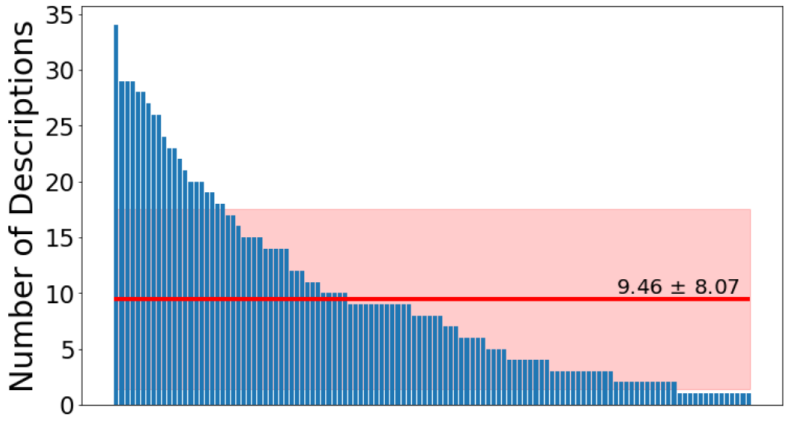


Figure 3.1: Number of Descriptions per Relation

Distribution of number of descriptions per relation, with decreasing frequency. The red line indicates the average number of descriptions per relation and the shaded area represents the standard deviation.

It could be possible that such an imbalance in number of descriptions can somehow bias the model. The investigation of how impactful this imbalance is could be a future line of work.

As additional information, each description has an average 7.37 ± 1.88 number of tokens[74].

We now turn to the creation of the actual (Generalised) Any-Shot Learning ((G)ASL) splits, which are themselves independent of the relations' descriptions.

[74] *As tokenized by spaCy (Honnibal and Montani, 2017, [31]), using the implementation of AllenNLP (Gardner et al., 2017, [22]).*

3.2 (Generalised) Any-Shot Learning Splits

For the design of our proposed splits we take as a strong basis the work of Xian et al. (2017, [73]), who actually proposed the *de facto* GZSL splits currently used by the CV community. However, before we jump into the actual splits, we further process the data made available by Levy et al. (2017, [37]).

3.2.1 Further Pre-Processing of UW-RE

From early experiments we learned that our method can be quite memory intensive. On top of that, the **UW-RE** dataset contains some very long sentences, that would only accentuate the problem further. As such, we start by only considering sentences that have a maximum tokenized[75] length of 60. This leaves us with 1.478.736 unique sentences, spanning a total of 2.376.807 unique relation mentions.

[75] *Using the same tokenizing method as before. Refer to margin note 74.*

In their work Obamuyide and Vlachos (2018, [46]) claim that they mask the relevant entities of each mention with **SUBJECT_ENTITY** and **OBJECT_ENTITY** to avoid having their method overfit particular entities. We follow the same approach. Note that this also allows the ESIM model to know which entities are relevant to the relation mention being classified.

We would also like to have an idea of how good our method is at estimating the presence of a relation given no specific annotation, as this could potentially pave the way for future unsupervised approaches.

As such, first we decided to ground each sentence with one specific relation mention, out of those present in the sentence (essentially assigning to the sentence the relation label of the selected mention). Specifically, we chose one of the mentions associated with the relation that has the fewest overall instances.

Second, for each of the sentences we create two other versions: an unannotated version[76] and another version, where we apply a generic mask for any entities found by a NER algorithm[77], like in the example below:

[76] *That is, just the plain sentence, with no masking whatsoever.*

[77] *We make use of AllenNLP's (Gardner et al., 2017, [22]) fine-grained NER algorithm.*

Relation: *author*

- Unmasked: *The Dream Room (German: Die Traumbude) was Erich Maria Remarque's first novel.*
- Subj/Obj-Masking: *SUBJECT_ENTITY (German: Die Traumbude) was OBJECT_ENTITY's first novel.*
- NER-Masking: *ENTITY (ENTITY: ENTITY) was ENTITY ENTITY novel.*

This gives us an aligned version of all the data, where each subject/object annotated sentence has a corresponding unmasked and NER-masked counterparts. We refer to any one instance as the tuple (unmasked, subject/object masking, NER-masking).

Besides allowing us to have an idea of the performance our method when given no specific mention annotation, this unmasked and NER-masked versions also allow us to test how much the model actually overfits on specific entities, as suggested by Obamuyide and Vlachos (2018, [46]).

It is important to note that the masking procedures, both the subject/object and the NER masking, can reduce different, but similar, sentences to an equal surface form, as can be seen below:

- Sentence 1: *The Scot Abroad is a book by John Hill Burton.*
- Sentence 2: *The Longest Whale Song is a book by Jacqueline Wilson.*
- Masked Version: *SUBJECT_ENTITY is a book by OBJECT_ENTITY.*

In order to avoid potentially having the same sentence in multiple, supposedly disjoint, splits (train/validation/test), we keep only those aligned tuples (unmasked, subject/object masking, NER-masking) that correspond to the relation that has the fewest overall instances. This gives us a final 996.361 unique sentences and mentions. Assigning them to the relation with the fewest overall instances helps in mitigating the data imbalance problem, but, nonetheless, it is still acutely present, as can be seen in figure 3.2.

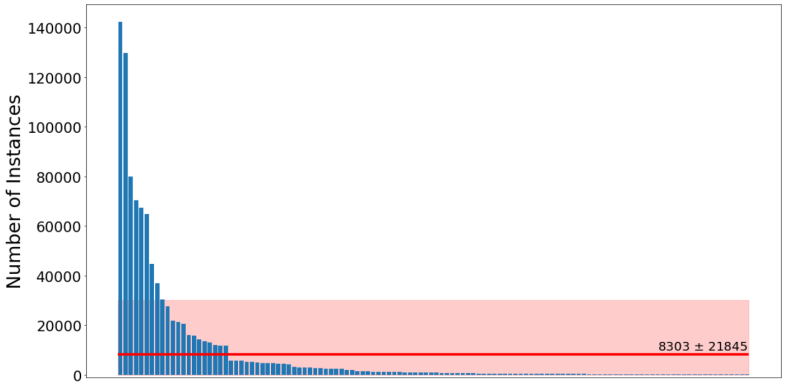


Figure 3.2: Number of Instances per Relation

Distribution of number of instances per relation, with decreasing frequency. The red line indicates the average number of descriptions per relation and the shaded area represents the standard deviation.

One important remark to be made is that both the grounding and masking procedures, described in this subsection, can lead to partially noisily labelled sentences. This happens due to the fact that the original sentences might have expressed more than one relation[78] or sentences that expressed different relations might have been reduced to the same surface

[78] *Under the Distant Supervision approach.*

form. While the label assigned to the (unmasked, subject/object masking, NER-masking) aligned pair is correct, it might not have been unique. This is hardly relevant for the subject/object case (happens with only $\approx 2\%$ of the instances), as the annotation of the entities makes the assigned label to be more in line with the true one. For the other unmasked and NER-masking types, this is more accentuated ($\approx 40\%$), as these cases span the entire sentence, and not only specific mentions. Nonetheless, as was previously stated, the purpose of the unmasked and NER-masked versions is just to allows us to have an idea of how the method performs when no annotation is given and to test for entity overfitting.

All in all, as the main purpose is to build (G)ASL splits for RC and the annotated part of our data[79] is barely more noisy than the data resulting from a Distant Supervision labelling approach, we use these 996.361 unique sentences/mentions to build our (G)ASL splits.

[79] *Which is the one used for actual RC.*

3.2.2 (G)ASL Splits Description

In this subsection we explain the actual creation of our (G)ASL splits. We will attempt to cover each important aspect of them in a structured way, so as to make the motivation and steps taken behind each one as clear as possible.

***Seen* Classes vs *Unseen* Classes**

Any (G)ASL setting will have some associated *seen* and *unseen* classes. Since the bulk of the training data consists of instances that belong to the *seen* classes, the bulk of the test data belongs to the *unseen* classes and train splits usually need to be significantly larger than test splits, Xian et al. (2017, [73]) suggest using the least frequent classes as the *unseen*/test classes.

Accordingly, we threshold our *seen*/*unseen* relations at 1.000 instances. This gives us 55 *seen* relations and 65 *unseen* relations, corresponding to 970.866 and 25.495 instances, respectively.

Evaluation Settings

Although we are proposing evaluation splits for both ASL and GASL settings, it might also be beneficial to have an evaluation setting for a normal classification scenario, i.e. where there is no distinction between *seen* and *unseen* classes. We call this a Normal Learning (NL) setting.

Therefore, our proposed evaluation splits cover a Normal Learning (NL) setting, a Zero-Shot Learning (ZSL) setting, a Generalised Zero-Shot Learning (GZSL) setting, Few-Shot Learning (FSL) settings and Generalised Few-

Shot Learning (GFSL) settings. Concerning the (G)FSL settings we consider Shot values[80] of 1, 2, 5 and 10. That is, we consider (G)FSL settings where the *unseen* classes have, at training time, 1, 2, 5 and 10 instances, respectively. These Shot values allow for an evaluation of the impact of increasing the available number of instances of the *unseen* classes, without overburdening the number of experiments to be executed.

[80] *We identify each particular (G)FSL setting' Shot number by abbreviating it to (G)FSL-M, where M is the Shot number.*

Regularisation

Regularisation is an important aspect of ML approaches, making it harder for models to overfit the data they are trained with and, consequently, usually providing better generalisation capabilities. One possible way to regularise models is through early stopping.

With early stopping one usually has an additional held out split, a validation split, that is used to check when a model starts overfitting. During the beginning of the training procedure the model learns to fit the training data better, and both the training and validation losses[81] go down. However, at some point, the model might start fitting the training data too well. This usually means that it has started tuning in on the noisy characteristics of the specific training data it is shown. As such, it will not generalise as well to other data. With a validation set one can check for this phenomena, as the validation loss will most likely stop decreasing[82] (and will probably actually start increasing), as opposed to the training loss, which will keep decreasing. This difference provides an indication of when the training of the model should be stopped, due to overfitting.

[81] *Or some other distinct metric improves.*

[82] *Or some other distinct metric will stop improving.*

Given its simplicity as a regularisation method, we decided to include validation splits for early stopping. Consequently, we will have train splits, validation splits and test splits.

Statistical Analysis

We would also like to be able to provide a structured way of performing a statistical analysis of obtained results. We feel that this can be readily achieved by evaluating models over multiple disjoint folds. Accordingly, we associate each evaluation setting with a specific set of folds, thus allowing us to have a statistical evaluation of methods' performance on any particular setting. A more detailed description of the number of folds will be provided further on.

Class Coverage and Robustness

In Normal Learning (NL) settings, folds are used not only for Hyperparameter Tuning, but also to test a model's robustness against the inherent difficulty of instances, both at training time and evaluation time. This is done to avoid uniquely training and evaluating a model on data that could

be potentially easier than what would be usual for the true data distribution.

One interesting aspect is that when it comes to the classes that are present in each (fold, split) pair, most (G)ASL works follow a similar approach as to that done on NL settings. For example, in NL settings, the classes present in the splits of any fold are the same classes present in the splits of any other fold. That is, regardless of the fold, the model will be trained on classes A, B and C and evaluated on classes A, B and C.

On the other hand, this is not the case for (G)ASL settings, where the aim is to train on some classes, say A and B, and evaluate on a disjoint set of classes, say C. Nonetheless, similarly to NL settings, across folds most (G)ASL works still train on the same classes, A and B, and equally evaluate on the same classes, C. That is, if on fold 1 the model is trained on classes A and B, and evaluated on class C, then on fold 2 it will also be trained on classes A and B and evaluated on class C.

We argue that this is not an ideal evaluation scenario for (G)ASL settings. Consider the following scenario:

- As *seen* classes we have the classes Beetle, Tuna, Falcon and Dog.
- As *unseen* classes we have the classes Wasp, Vulture, Wolf and Salmon.
- *Seen* classes are used for training and *unseen* classes for evaluation.

As the point of (G)ASL methods is to train a model that can generalise to previously *unseen* classes, as discussed earlier, it would not be a big surprise if a model trained under the scenario above would perform reasonably well. In fact, the test classes have a strong corresponding class in the train classes. For example, dogs and wolves are not so different. Similarly, given that Tuna and Salmon are the only fish in the *seen* and *unseen* sets, it would probably be relatively easy for the model to distinguish the class Salmon at test time.

But how well would the model perform if the train set did not have the class dog, for instance? Or if the test set did not have the class Salmon? In the first case, the model would not benefit from having seen a class with features very close to that of a wolf, which would potentially impair its ability to recognise a wolf. Likewise, in the second case it could be that the class Tuna would actually act as a distractor during training time, pushing the model to learn how to recognize features that would not be important at test time.

We propose a way to address the testing of a model's robustness against such scenarios, where one does not know which training classes will be relevant for the model's evaluation or which evaluation classes benefit from specific training classes.

In order to achieve that, we design each fold to be independent from the other folds, within the same (G)ASL setting, by making each fold have unique train, validation and test splits.

This allows us to assign a class configuration for the train, validation and test splits that is different for each fold, thus supporting class robustness testing, to some degree. We achieve this by subsampling, for each (fold, split) pair, the classes that will be present in that (fold, split) pair. For example, consider we have the following class sets:

	Set 1	Set 2
Classes	A, B, C, D, E	X, Y, Z

Table 3.2: Class Sets Example

where Set 1 is used for the training splits and Set 2 is used for the evaluation[83] splits. Now imagine that we have two folds for a specific (G)ASL setting. We could subsample the classes for each (fold, split) in the following way, for example:

[83] *Validation and test splits.*

	Split		
	Train	Validation	Test
Fold 1	A, C, D	X	Z
Fold 2	B, C, E	Z	Y

Table 3.3: (Fold, Split) Class Subsampling Example

This way the model is trained on different classes and also evaluated on different classes, which gives us a better perspective of how good the model is at extrapolating knowledge from a generic set of *seen* classes to a generic set of *unseen* classes.

Overall, we expect that the performance of models, whatever the evaluation metric, will have a higher variance across folds, due to some folds probably having harder classes than others.

Although this framework can potentially give a better idea of the generic generalisation capability of a model, some considerations should be taken when designing it. In particular, for each fold, all splits, train, validation and test, should have disjoint subsets of the available classes (disregarding the classes present due to FSL and the Generalised case). Additionally, it is important to guarantee that the class distribution across folds is as even as possible. For example, if we have 10 folds and class A gets sampled on 9 of those folds, while class B gets sampled on only 2 of those folds, we would be introducing an inherent evaluation bias towards the features of class A. Who's to say that the model would not perform better or worse, on average, if the features of class A were not always present, or if, alternatively, the features of class B were available more frequently?

In line with the reasoning above, we suggest ensuring a class distribution over folds that is as uniform as possible, while still making the subset of classes, present on any (fold, split) pair, random. In our case, we enforce this by ensuring that, *for each split, across folds*, the class that occurs in the most folds occurs at most one time more than the class that occurs the least. That is, if class A is the class that occurs the most times on the train splits[84], because it got sampled 4 times (on folds 1, 3, 5, and 6 , for example (out of 10 folds)), then any other class, let's say B, gets sampled on at least 3 train splits (on folds 5, 8, 9, for example).

[84] *We use the train splits as an example, but the procedure generalises for the validation and test splits.*

This method of undersampling classes over (folds, splits) has an additional advantage, besides helping with testing a method's class robustness. In particular, this means that classes occur in fewer folds, for each setting, which makes it easier to build the folds. This is the case, since if classes appeared in all folds, some of them would not have enough instances to be present in all of them, while ensuring that instances in one fold are not present in another fold. For example, for FSL-10, if we have 10 folds, then, since folds are disjoint at the instance level[85], we would need 100 instances for the train splits alone. If we also add the instances required for the evaluation splits, then we would need even more than that. This can become a problem, as some classes have fewer than 100 instances.

[85] *By our design.*

Dataset Types

One of the main points argued by Xian et al. (2017, [73]) is that before test time, models should never see, in any way, any instances (besides the Few-Shot ones) of the *unseen* classes. In their work they stress this point, as there had been some previous works that had performed hyperparameter tuning based on evaluation on the *unseen* classes, which goes against the mentality behind (G)ASL settings.

Additionally, in CV, a lot of feature extraction models make use of ImageNet (Deng et al., 2009, [16]), a CV classification dataset that is also what is usually used for testing (G)ASL approaches. Xian et al. (2017, [73]) argue that the feature extraction models should also be trained uniquely on the *seen* classes, so that no information regarding the *unseen* classes is potentially leaked before testing. We highlight this point for the benefit of future NLP research. If one is to learn, or finetune, an embedding model and perform (G)ASL tasks, then one should be careful to make sure that only the *seen* classes are used. On the other hand, if one uses a pretrained embedding model, such as ELMo (Peters et al., 2018, [51]) or BERT (Devlin et al., 2018, [17]), this might be unavoidable. However, it might also not be impactful, since these models are, most likely, not trained on data specifically structured for the particular task that is to be evaluated.

For these reasons we create two distinct main datasets and, for a matter of convenience, we also make available a tiny debugging dataset:

- Debugging (DEBUG): A tiny dataset that makes use of only a small subset of the *seen* classes and that has extremely few instances and folds. It is meant for debugging training procedures, models and their evaluation. Concretely:
  - It only covers the NL, (G)ZSL and (G)FSL-1 settings.
  - Only 9 classes are considered, always the same across settings. These have been chosen at random from the set of *seen* classes.

- Hyperparameter Tuning (HT): A small dataset, with a few folds per setting, that only has data concerning the *seen* classes. It is meant to be used for both hyperparameter tuning and/or the finetuning of feature embedding models. Concretely:
  - It covers all settings (NL and (G)ASL).
  - Only *seen* classes are considered. Classes are subsampled, for each split, at random from all *seen* classes. Disjointness, between classes, is ensured across the splits of any fold.

- Final Evaluation (FE): A large dataset that has a significant number of folds, per setting, and has data concerning both *seen* and *unseen* classes. It is meant to be used for proper model evaluation. Concretely:
  - It covers all settings (NL and (G)ASL).
  - For train splits only *seen* classes are considered. For validation and test splits only *unseen* classes are considered. Disjointness, between classes, is also ensured for these 2 splits, for each fold.

The exact number of folds, classes per split and the number of instances per split, for each dataset, can be found below:

Dataset Type	**DEBUG**			**HT**			**FE**		
# Folds	2			5			10		
Split	Train	Val	Test	Train	Val	Test	Train	Val	Test
# Classes	5	2	2	25	10	10	50	20	20
# Instances	20	10	10	10000	500	1000	50000	625	1250

Table 3.4: Datasets Characteristics

Final Properties and Considerations

There are some other properties of the proposed splits that are worth mentioning.

First, the NL setting always makes use of all classes available, for all the splits. For example, consider the HT dataset. In this case all splits (train, validation and test) have instances of all the available classes. That is, each split has instances of the 55 *seen* classes. Likewise, for the FE dataset, each split has instances of the 120 existing classes.

Second, the instances present in any split are sampled according to the relative data distribution of the classes considered in that split. This excludes (G)FSL *unseen* training instances, which are randomly picked from the corresponding classes at the start, while the remaining number of instances of the *seen* classes is sampled according to the relative data distribution method just described[86]. The Generalised evaluation splits are also excluded. For these, we assign 50% of the available number of instances to the *seen* classes, and the remaining 50% to the *unseen* classes. Then, for each separate *seen*/*unseen* set, the corresponding number of instances is sampled according to the relative data distribution method, considering only the classes, of the respective set, that have been sampled for the corresponding splits. For HT only *seen* classes are considered. The equivalent[87] *seen* classes correspond to the classes sampled for the train split and the equivalent *unseen* classes correspond to the classes sampled for whichever evaluation split is being considered, either validation or test.

[86] *Considering only the relative data distribution of the* seen *classes.*

[87] *What we mean by this is to draw an analogy to the FE dataset, that actually makes use of the* unseen *classes.*

Third, for the train splits of the NL setting we ensure at least 1 instance of each class, thus following as much as possible the observed data distribution, while still guaranteeing the presence of all classes. On the other hand, for the (G)ASL settings, things are slightly different. For the train splits, we guarantee a minimum of 25 instances for the *seen* classes, thus creating a clear distinction between *seen* and *unseen* classes. As for the evaluation splits, for both NL and (G)ASL settings we guarantee at least 5 instances of each class. This 5 instances minimum also applies to *seen* classes present in evaluation splits, due to the Generalised case.

Fourth, the HT dataset is completely disjoint at the instance level, even across settings. That is, for an instance x:

$$x \in (setting\_i, fold\_j, split\_k) \Rightarrow x \notin (setting\_i', fold\_j', split\_k')$$
$$\text{such that: } i \neq i' \vee j \neq j' \vee k \neq k'$$

Fifth, for the FE dataset, each setting has completely disjoint train and validation splits, at the instance level, but settings are not disjoint:

$$x \in (fold\_j, split\_k) \Rightarrow x \notin (fold\_j', split\_k')$$
$$\text{such that: } (j \neq j' \vee k \neq k') \wedge split \in \{train, val\}$$

On the other hand, for FE, instances in any of the test splits can also be in the test splits of other folds. Furthermore, each fold has completely disjoint splits, at the instance level.

Finally, instances that belong to the *seen* classes of the train splits of the FE dataset can also be present in the HT dataset. That is, the instances of the HT dataset can be reused to build the train splits of the FE dataset. However, we ensure that they are not present in any evaluation (validation/test) split of the FE dataset.

We hope that this proposed splits provide a more realistic evaluation scenario for (G)ASL settings, so as to render comparison of future research methods in NLP, and in particular RC, more realistic. We would like to highlight that even though we include non-Generalised settings, such as ZSL or FSL settings, we do so only as a way of providing a comparison to their Generalised counterpart. We hope to, this way, show how much easier the non-Generalised cases are.

We make our proposed splits available, under the MIT License, at `https://github.com/Nuno-Mota/Generalised-Any-Shot-Learning---Relation-Classification-dataset`.

Now that a complete description of the proposed splits has been provided, let us turn our attention to our evaluated models and the experiments we perform on these splits.

Chapter 4

Method

In this chapter we discuss the more practical aspects of this thesis. Concretely, in section 4.1 we present the models that we train and evaluate on the previously introduced (G)ASL splits. In section 4.2 we examine the metrics that are used to assess the performance of our models. We also introduce[88] a small extension of the existing Macro F1-Score metric, the Harmonic Macro F1-Score, which is specifically targeted at scenarios where both *seen* and *unseen* classes exist. Finally, in section 4.3 we discuss several insights gathered from the experiments ran on the new (G)ASL splits.

[88] *In CV literature the Harmonic Mean between the top-1 per class Accuracy of the* seen/unseen *sets is used: Here we simply propose using instead the Macro F1-Score of the* seen/unseen *sets.*

4.1 Models

As we argued in the beginning of this thesis, one of the things we aimed at, was determining to what extent more complex NLI models were beneficial for (G)ASL tasks, when compared against much simpler approaches.

Accordingly, in 4.1.1 we start by presenting an extremely simple model capable of performing (G)ASL tasks. Afterwards, in 4.1.2, we discuss in depth the ESIM model of Chen et al. (2017, [11]). Finally, in section 4.1.3 we discuss how to perform a categorical estimation over all relevant classes, how we train the models and how the testing phase differs from the training phase.

4.1.1 BiLSTM Baseline

Our first and simplest model is nothing more than a Bidirectional LSTM (BiLSTM) (Graves and Schmidhuber, 2005, [24]). While we assume that the reader is familiarised with how a LSTM (Hochreiter and Schmidhuber, 1997, [30]) works, we will provide a high level overview.

Simply stated, the key factor of a LSTM is that it maintains an internal representation of an input sequence, $\boldsymbol{z} = (\boldsymbol{z_1}, \boldsymbol{z_2}, \ldots, \boldsymbol{z_{L_z}})$, where each element of $\boldsymbol{z}$ has an associated vector representation[89]. This representation is called the *cell state*, $\boldsymbol{c^z}$, and it is what enables LSTMs to encode long term dependencies, an ability which normal RNNs fail to achieve.

[89] *That can either be a one-hot encoding, a vector from a learned embedding matrix or a pretrained word embedding.*

Besides the cell state, at each step a LSTM reads one more element of the sequence $\boldsymbol{z}$, $\boldsymbol{z_t}$, and projects this new element to a hidden representation, $\boldsymbol{h_t^z}$. This new hidden representation depends not only on the corresponding

new sequence element, $\boldsymbol{z_t}$, but also on the hidden representation of the element that came immediately before, $\boldsymbol{h^z_{t-1}}$, and the sequence's long term dependencies encoded, up to that step, in the cell state, $\boldsymbol{c^z_{t-1}}$. The usage of the immediate previous state provides the Short Term memory aspect of LSTMs. Mathematically (and ignoring the inner workings of a LSTM):

$$\begin{aligned} \boldsymbol{c^z_0} &= \boldsymbol{0} \\ \boldsymbol{h^z_0} &= \boldsymbol{0} \\ (\boldsymbol{c^z_t}, \boldsymbol{h^z_t}) &= \text{LSTM}\left(\boldsymbol{z_t}, \boldsymbol{c^z_{t-1}}, \boldsymbol{h^z_{t-1}}; \boldsymbol{W}\right) \end{aligned} \tag{4.1}$$

where $\boldsymbol{W}$ represents the full set of parameters of the LSTM.

On the other hand, a BiLSTM is nothing more than two distinct LSTMs, where one reads the sequence from beginning to end (also known as Forward LSTM and identified with $\rightarrow$) and the other reads the sequence from end to beginning (also known as Backward LSTM and identified with $\leftarrow$). These can be represented mathematically as:

$$\begin{aligned} \overrightarrow{\boldsymbol{c^z_0}} &= \boldsymbol{0} \\ \overrightarrow{\boldsymbol{h^z_0}} &= \boldsymbol{0} \\ (\overrightarrow{\boldsymbol{c^z_t}}, \overrightarrow{\boldsymbol{h^z_t}}) &= \text{LSTM}\left(\boldsymbol{z_t}, \overrightarrow{\boldsymbol{c^z_{t-1}}}, \overrightarrow{\boldsymbol{h^z_{t-1}}}; \overrightarrow{\boldsymbol{W}}\right) \end{aligned} \tag{4.2}$$

and

$$\begin{aligned} \overleftarrow{\boldsymbol{c^z_{L_z+1}}} &= \boldsymbol{0} \\ \overleftarrow{\boldsymbol{h^z_{L_z+1}}} &= \boldsymbol{0} \\ (\overleftarrow{\boldsymbol{c^z_t}}, \overleftarrow{\boldsymbol{h^z_t}}) &= \text{LSTM}\left(\boldsymbol{z_t}, \overleftarrow{\boldsymbol{c^z_{t+1}}}, \overleftarrow{\boldsymbol{h^z_{t+1}}}; \overleftarrow{\boldsymbol{W}}\right) \end{aligned} \tag{4.3}$$

In NLP, BiLSTMs are very useful in producing context aware word embeddings, but they are also often employed in the construction of overall sentence embeddings. Usually, such embeddings leverage the last hidden state of both directions.

For example, for the sentence $\boldsymbol{z}$ of length L_z, the corresponding sentence embedding $\boldsymbol{h^z}$, can be constructed as:

$$\boldsymbol{h^z} = \left[\overrightarrow{\boldsymbol{h^z_{L_z}}}; \overleftarrow{\boldsymbol{h^z_1}}\right] \tag{4.4}$$

where $[\cdot;\cdot]$ represents a concatenation.

Figure 4.1 provides an overview of the entire process:

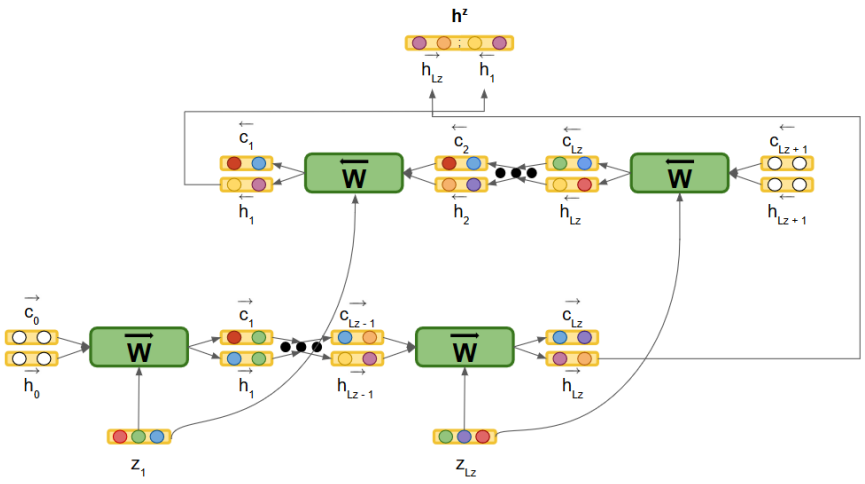


Figure 4.1: Diagram of BiLSTM Sentence Embedding

For our task we need to compare two input sentences, just like in NLI. The common practice in NLI is two use two distinct BiLSTMs when producing sentence[90] embeddings, one for the *premise*, $\boldsymbol{p}$, and another for the *hypothesis*, $\boldsymbol{h}$. This is also the case for the ESIM model, specifically its first component, the input encoding component. For our baseline we choose to have a model that is similar to this first component of ESIM, as this allows us to assert the benefit of adding more complexity to the model, as previously discussed.

[90] *Or word-level hidden representations.*

However, while the use of two distinct BiLSTMs might allow each sentence to have a more meaningful hidden representation on its own, it also makes that task of matching the two distinct representations harder. In our preliminary experiments (4.3.2) we explore this topic further. For now, though, it suffices to say that due to the difficulty of matching the representations produced by each individual BiLSTM, we chose to employ a single BiLSTM[91] to encode , as per equation 4.4, both our sentence containing a relation mention, $\boldsymbol{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_{L_x}})$, and a candidate relation description $\boldsymbol{y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_{L_y}})$:

[91] *This is also the case for ESIM's first component, which means we can still compare the addition of complexity to the model.*

$$\boldsymbol{h^x} = \left[\overrightarrow{\boldsymbol{h^x_{L_x}}}; \overleftarrow{\boldsymbol{h^x_1}}\right]$$

$$\boldsymbol{h^y} = \left[\overrightarrow{\boldsymbol{h^y_{L_y}}}; \overleftarrow{\boldsymbol{h^y_1}}\right]$$

Finally, we estimate a score of how compatible the sentence $\boldsymbol{x}$ is with description $\boldsymbol{y}$, by computing the inner product between $\boldsymbol{h^x}$ and $\boldsymbol{h^y}$. That

is, aligning the formalism used in the previous chapters:

$$F(x, y; \boldsymbol{W}) = \boldsymbol{h}^{\boldsymbol{x}} \cdot \boldsymbol{h}^{\boldsymbol{y}} \tag{4.5}$$

where $\boldsymbol{W} = \overrightarrow{\boldsymbol{W}} \bigcup \overleftarrow{\boldsymbol{W}}$, takes into account the parameters of both the Forward and Backward LSTMs.

This is an extremely simple model, that hopefully is capable of encoding, for both the sentence and the relation description, the underlying relation. If $\boldsymbol{x}$ represents relation r, then the idea is that $\boldsymbol{h}^{\boldsymbol{x}}$ will be a distinctive hidden representation of the underlying relation. Likewise, the candidate description of a specific relation r, $\boldsymbol{y_r}$, should have a similar hidden representation $\boldsymbol{h}^{\boldsymbol{y_r}}$. Ideally we would have $\boldsymbol{h}^{\boldsymbol{x}} \approx \boldsymbol{h}^{\boldsymbol{y_r}}$ and $\boldsymbol{h}^{\boldsymbol{x}} \not\approx \boldsymbol{h}^{\boldsymbol{y_{r'}}}$ for any other relation r'.

Can this model be too simple, though? Quite possibly, which leads us to ESIM, a significantly more complex model that is meant to be better at identifying the similarities and differences between $\boldsymbol{x}$ and a candidate relation description $\boldsymbol{y}$.

4.1.2 ESIM Set to Set

The Enhanced Sequential Inference Model (ESIM) model of Chen et al. (2017, [11]), is a proven architecture in NLI, that has yielded impressive results with relatively "few"[92] parameters.

[92] *With around 4 million parameters, compared to models with 40 million and even 300 million parameters.*

Chen et al. (2017, [11]) defined three main components of their model. Concretely, they introduced what they call the *input encoding* component, the *local inference modelling* component and, finally, the *inference composition* component. We shall now address each one of them. Note that the model of Chen et al. (2017, [11]) also takes into consideration syntactic parse trees, but we do not make use of such aspect of the model and, as such, we do not discuss it.

Input Encoding

The input encoding part of ESIM consists of BiLSTMs, which are used to produce context aware word embeddings, with the size of the BiLSTMs' hidden dimension. While our previous model, the BiLSTM baseline, also makes use of a BiLSTM, there it is used to produce overall sentence embeddings, as opposed to the BiLSTMs of ESIM, which produce word embeddings.

When using BiLSTMs to produce word embeddings, the hidden representation of each word is the concatenation of the corresponding hidden representations produced in both directions, as in equations 4.2 and 4.3. For

example, the hidden representation of word at step t, of a generic sentence $\boldsymbol{z} = (\boldsymbol{z_1}, \boldsymbol{z_2}, \ldots, \boldsymbol{z_{L_z}})$ is:

$$\boldsymbol{h_t^z} = \left[\overrightarrow{\boldsymbol{h_t^z}}; \overleftarrow{\boldsymbol{h_t^z}}\right] = \text{BiLSTM}\left(\boldsymbol{z}, t; \boldsymbol{W}\right) \tag{4.6}$$

The following figure aims to illustrate this:

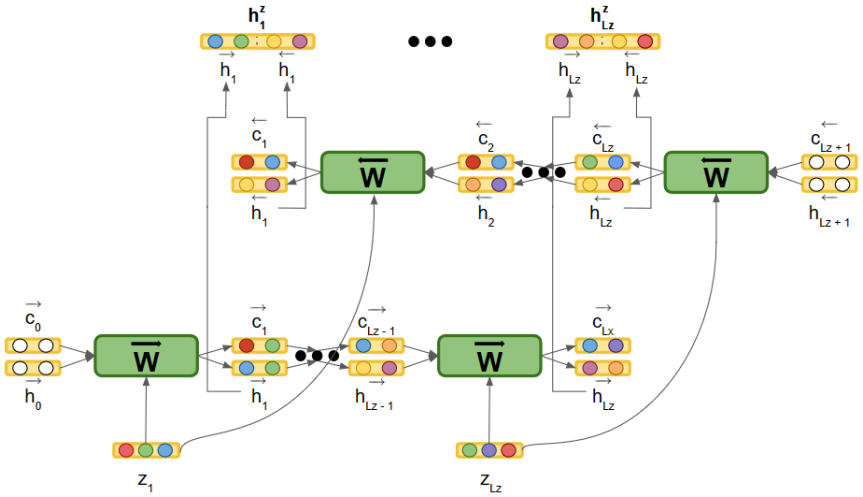


Figure 4.2: Diagram of BiLSTM Hidden States

Example of the construction of the hidden states of an arbitrary sequence z, using a BiLSTM.

As ESIM is designed for the task of NLI, it takes as input two sentences, a *premise* $\boldsymbol{p}$ and an *hypothesis*, $\boldsymbol{h}$, to which we shall, henceforth, simply refer to as $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. Chen et al. (2017, [11]) make use of two distinct BiLSTMs, one for $\boldsymbol{x}$ and one for $\boldsymbol{y}$, in order to produce context aware word-level hidden representations for each of them.

However, similarly to our previous discussion for the BiLSTM baseline, this makes it harder to compare representations produced by one of the BiLSTMs against representations produced by the other. As such, for the ESIM case we also tested the model using two distinct BiLSTMs, one for the sentence $\boldsymbol{x}$, and the other for the relation description $\boldsymbol{y}$, or simply using a single BiLSTM for both. Again, the single BiLSTM formulation yielded better results.

Accordingly, the input component of our ESIM model consists of a single BiLSTM that is used to produce hidden word embeddings for both the sentence $\boldsymbol{x}$ and the relation description $\boldsymbol{y}$, such that:

$$\begin{aligned} \bar{\boldsymbol{x}} &= (\bar{\boldsymbol{x}}_1, \bar{\boldsymbol{x}}_2, \ldots, \bar{\boldsymbol{x}}_{L_x}) = \left(\boldsymbol{h_1^x}, \boldsymbol{h_2^x}, \ldots, \boldsymbol{h_{L_x}^x}\right) \\ \bar{\boldsymbol{y}} &= \left(\bar{\boldsymbol{y}}_1, \bar{\boldsymbol{y}}_2, \ldots, \bar{\boldsymbol{y}}_{L_y}\right) = \left(\boldsymbol{h_1^y}, \boldsymbol{h_2^y}, \ldots, \boldsymbol{h_{L_y}^y}\right) \end{aligned} \tag{4.7}$$

Local Inference Modelling

As we argued before, the understanding of whether an hypothesis $\boldsymbol{y}$ can be inferred from the context of a given premise $\boldsymbol{x}$, largely depends on the individual elements of both and, particularly, how these relate between themselves. Chen et al. (2017, [11]) leverage this understanding to their benefit and employ the intra-sentence attention proposed by Parikh et al. (2016, [48]).

The main difference between both works, in this particular aspect, is that Parikh et al. (2016, [48]) apply their intra-sentence approach directly on the word embeddings of $\boldsymbol{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_{L_x}})$, and $\boldsymbol{y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_{L_y}})$. On the other hand, Chen et al. (2017, [11]) apply it to the context-aware word embeddings produced by their input encoding component, $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{y}}$.

The attention mechanism itself does nothing more than a soft-alignment, at the word level, between both input sentences. Concretely, for our $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{y}}$ sentences, the attention-weights matrix $\boldsymbol{A} \in \mathbb{R}^{L_x} \times \mathbb{R}^{L_y}$ is computed as:

$$\boldsymbol{A_{ij}} = \bar{\boldsymbol{x}}_i \cdot \bar{\boldsymbol{y}}_j \tag{4.8}$$

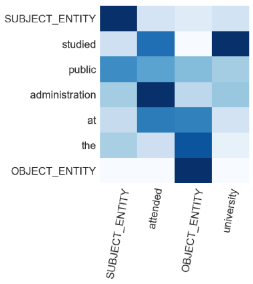


Figure 4.3: Attention Weights

Adapted from Obamuyide and Vlachos (2018, [46]).

In turn, these weights allow the composition of attention-weighed representations of $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{y}}$, respectively $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$, which are composed in respect to each other. Effectively, each attention-weighed word representation is computed by:

$$\begin{aligned} \hat{\boldsymbol{x}}_i &= \sum_{j=1}^{L_y} \frac{\exp\left(\boldsymbol{A_{ij}}\right)}{\sum_{j'=1}^{L_y} \exp\left(\boldsymbol{A_{ij'}}\right)} \bar{\boldsymbol{y}}_j \\ \hat{\boldsymbol{y}}_j &= \sum_{i=1}^{L_x} \frac{\exp\left(\boldsymbol{A_{ij}}\right)}{\sum_{i'=1}^{L_x} \exp\left(\boldsymbol{A_{ij'}}\right)} \bar{\boldsymbol{x}}_i \end{aligned} \tag{4.9}$$

Intuitively, for sentences $\boldsymbol{x}$ and $\boldsymbol{y}$, one can regard the attention-weighed vector $\hat{\boldsymbol{x}}$ as a representation that contains the elements of $\boldsymbol{y}$ that are relevant given $\boldsymbol{x}$ (the same holds from $\boldsymbol{y}$ to $\boldsymbol{x}$).

Chen et al. (2017, [11]) also propose a series of combinations between the context-aware hidden embeddings, $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{y}}$, and their respective attention-weighed counterparts, $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$. Specifically, they compute the difference and element-wise product, for each word, which are then concatenated with the original context aware vectors and the original attention-weighed vec-

tors:

$$\begin{aligned}
\boldsymbol{m}_t^x &= [\bar{\boldsymbol{x}}_t; \hat{\boldsymbol{x}}_t; \bar{\boldsymbol{x}}_t - \hat{\boldsymbol{x}}_t; \bar{\boldsymbol{x}}_t \odot \hat{\boldsymbol{x}}_t] \\
\boldsymbol{m}^x &= \left(\boldsymbol{m}_1^x, \boldsymbol{m}_2^x, \ldots, \boldsymbol{m}_{L_x}^x\right) \\
\boldsymbol{m}_t^y &= [\bar{\boldsymbol{y}}_t; \hat{\boldsymbol{y}}_t; \bar{\boldsymbol{y}}_t - \hat{\boldsymbol{y}}_t; \bar{\boldsymbol{y}}_t \odot \hat{\boldsymbol{y}}_t] \\
\boldsymbol{m}^y &= \left(\boldsymbol{m}_1^y, \boldsymbol{m}_2^y, \ldots, \boldsymbol{m}_{L_y}^y\right)
\end{aligned} \tag{4.10}$$

The authors argue that such combination of vectors helps in sharpening the inference information contained between the two sentences, allowing for more easily capturing inference relationships, such as contradiction.

Inference Composition

Now that each word of $\boldsymbol{x}$ is aware of not only its own context, but also the inference relevant elements of $\boldsymbol{y}$ (the same holds from $\boldsymbol{y}$ to $\boldsymbol{x}$), Chen et al. (2017, [11]) propose to further improve the overall inference awareness of each word.

In order to achieve that, they recourse once more to BiLSTMs, which they use to produce new word embeddings. The major difference is that these new word embeddings are not only aware of inference properties directly related to themselves, but also, after the use of the BiLSTM, the inference properties related to the other words in their own sentence.

However, before producing these new vectors they reduce the dimensionality of the vectors produced in the previous step, $\boldsymbol{m}_x$ and $\boldsymbol{m}_y$, which, due to the concatenation of four different components increase the overall complexity. In their work this is achieved by function $G(\cdot)$, a single layer MLP, with a Rectified Linear Unit (ReLU) activation function (Glorot et al., 2011, [23]):

$$\begin{aligned}
\bar{\boldsymbol{m}}_t^x &= \text{BiLSTM}\left(G\left(\boldsymbol{m}^x\right), t; \boldsymbol{W}\right) \\
\text{with} \quad G\left(\boldsymbol{m}^x\right) &= \left(G\left(\boldsymbol{m}_1^x\right), G\left(\boldsymbol{m}_2^x\right), \ldots, G\left(\boldsymbol{m}_{L_x}^x\right)\right) \\
\bar{\boldsymbol{m}}_t^y &= \text{BiLSTM}\left(G\left(\boldsymbol{m}^y\right), t; \boldsymbol{W}\right) \\
\text{with} \quad G\left(\boldsymbol{m}^y\right) &= \left(G\left(\boldsymbol{m}_1^y\right), G\left(\boldsymbol{m}_2^y\right), \ldots, G\left(\boldsymbol{m}_{L_y}^y\right)\right)
\end{aligned} \tag{4.11}$$

In our particular case, we decided to replace the ReLU activation function with its corresponding leaky version, the Leaky Rectified Linear Unit

(LeakyReLU), as we found that there were a significant number of times where we were encountering dead neurons in $G(\cdot)$. The LeakyReLU avoids this problem by always having a gradient flow when backpropagating, which the normal ReLU does not.

Although at the moment we have a series of highly inference informative word embeddings, we still do not have an easy way of simply asserting that $\boldsymbol{x}$ entails $\boldsymbol{y}$. Thus, Chen et al. (2017, [11]) propose a final composition, in order to obtain a final representation of the sentence pair, as a whole. Specifically, they suggest, for each sentence, averaging and max pooling the word embeddings, over each embedding dimension. Specifically, assuming that $\forall_t\ \bar{\boldsymbol{m}}_t^x \in \mathbb{R}^d\ :\ t \in \{1, 2, \ldots, L_x\}$ and $\forall_t\ \bar{\boldsymbol{m}}_t^y \in \mathbb{R}^d\ :\ t \in \{1, 2, \ldots, L_y\}$:

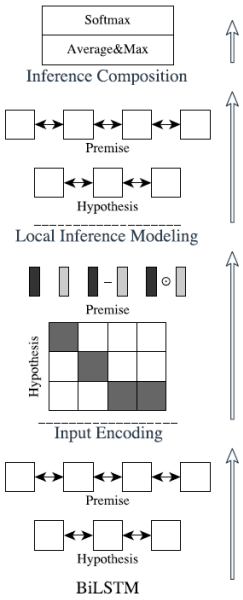


Figure 4.4: ESIM

Adapted from Chen et al. (2017, [11]).

$$
\begin{aligned}
\boldsymbol{v}_{ave}^x &= \sum_{t=1}^{L_x} \frac{\bar{\boldsymbol{m}}_t^x}{L_x} \\
\boldsymbol{v}_{max}^x &= \left[\max_{t=1}^{L_x} \bar{\boldsymbol{m}}_{t,1}^x,\ \max_{t=1}^{L_x} \bar{\boldsymbol{m}}_{t,2}^x,\ \ldots,\ \max_{t=1}^{L_x} \bar{\boldsymbol{m}}_{t,d}^x\right] \\
\boldsymbol{v}_{ave}^y &= \sum_{t=1}^{L_y} \frac{\bar{\boldsymbol{m}}_t^y}{L_y} \\
\boldsymbol{v}_{max}^y &= \left[\max_{t=1}^{L_y} \bar{\boldsymbol{m}}_{t,1}^y,\ \max_{t=1}^{L_y} \bar{\boldsymbol{m}}_{t,2}^y,\ \ldots,\ \max_{t=1}^{L_y} \bar{\boldsymbol{m}}_{t,d}^y\right]
\end{aligned}
\tag{4.12}
$$

These are composed, forming a single $(\boldsymbol{x}, \boldsymbol{y})$ representation vector:

$$
\boldsymbol{v} = [\boldsymbol{v}_{ave}^x, \boldsymbol{v}_{max}^x, \boldsymbol{v}_{ave}^y, \boldsymbol{v}_{max}^y] \tag{4.13}
$$

In NLI the task is to predict one of three classes: entailment, neutral or contradiction. Accordingly, Chen et al. (2017, [11]) give this final $(\boldsymbol{x}, \boldsymbol{y})$ representation, $\boldsymbol{v}$, as input to a 1-hidden layer MLP that projects it into a three dimensional vector, which, in turn, is turned into the parameters of a categorical distribution, through the usage of a *softmax* output layer. This is also the approach taken by Obamuyide and Vlachos (2018, [46]), except they turn it into a binary classification.

Unlike them, we do not want our method to perform the classification directly. Instead, we want the entirety of the ESIM model to simply compute a score of how likely it is that $\boldsymbol{y}$ can be entailed from $\boldsymbol{x}$. Consequently, we have this last MLP project $\boldsymbol{v}$ into a single scalar. Also unlike them, we use LeakyReLUs instead of regular ReLUs.

Let us now discuss how we turn this into a categorical classification over the possible relations.

4.1.3 Classification Over the Possible Relations

Now that we have defined our models' architectures, which, following the (G)ASL formalism, can be used as our $F(x, y; \boldsymbol{W})$ function, how exactly do we train them?

Before we address this topic, let us quickly review how the training and test procedures work for general NLI methods, including the work of Chen et al. (2017, [11]), or even how Obamuyide and Vlachos (2018, [46]) approach ASL using the same principles.

First, it is important to understand that a NLI training instance consists of the pair (p, h), and not either one of them individually. This makes it easy to simply draw instances ((p, h) pairs) into a mini-batch, compute, for each instance, the parameters of the previously mentioned 3-way[93] categorical, compute a loss and readjust the trainable parameters using any gradient based stochastic optimisation algorithm.

In the case of Obamuyide and Vlachos (2018, [46]) the process is very similar, as their approach consists of pairing each sentence $\boldsymbol{x}$, containing a relation mention, with several distinct relation descriptions $\boldsymbol{y}$, creating unique $(\boldsymbol{x}, \boldsymbol{y}_{r1})$, $(\boldsymbol{x}, \boldsymbol{y}_{r2})$, ... pairs[94], which become their new dataset. This way they can simply adapt the procedure above to their needs. That is, they simply turn the 3-way categorical into a binary classification and keep everything else the same, essentially performing multiple distinct binary classifications for the same $\boldsymbol{x}$.

Contrary to their approach, we want to directly perform a categorical estimation over all possible relations. Therefore, we need to somehow adapt the NLI procedure to one that suits our approach. As we want our method to be adaptable to new unforeseen situations, that is, to be able to recognize new classes "on the fly", an important aspect to note is that we cannot simply have a model that computes the parameters of a discrete distribution over a fixed support[95]. Fortunately for us, both of the models we have already discussed here, are either shaped, or have been adapted, in such a way that they can compute a score for each class, individually.

This is the core of our approach. Instead of pairing each instance $\boldsymbol{x}$ with specific candidate relation description $\boldsymbol{y}$, we keep our individual $\boldsymbol{x}$s as our instances, draw a minibatch $\mathcal{M} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$, where y_n represents the true label of $\boldsymbol{x}_n$, and compare each instance in the minibatch against all relations that are known to be present in the respective data split. The comparison between an instance $\boldsymbol{x}$ and the possible relations is done in such a way that for any two relations, r_1, r_2, the comparison of $\boldsymbol{x}$ with r_1 is completely independent of the comparison between $\boldsymbol{x}$ and r_2.

However, remember that we are not actually directly comparing $\boldsymbol{x}$ with

[93] *Remember the task is to classify each pair into one of 3 classes: entailment, neutral or contradiction.*

[94] *We noticed that this led them to treat the pairs as unique instances, which led to the presence of several $\boldsymbol{x}$ sentences in both training and evaluation splits. We argue that while this make sense for NLI, where the hypothesis h are indeed unique, it does not make as much sense for this specific case, as the $\boldsymbol{y}$s are identifying classes and are not themselves instances.*

[95] *That is, a categorical of fixed dimensions.*

relations, but with their descriptions. Since each relation can have multiple descriptions, what's the optimal way to use those descriptions?

We start by describing the evaluation procedure, as it is slightly simpler than the training one and helps pave the discussion for the latter.

Since language can be quite ambiguous, just as was discussed in 3.1.2, it makes sense that some relation descriptions are more apt than others for specific sentences, even within the same canonical relation. For example, for relation r a description d_1 might express sentence $\boldsymbol{x}$ better than another description d_2. Following this line of thought, we choose to, at evaluation time, compare each sentence with all the descriptions of each of the possible relations of the split. This allows us to, for each relation, pick the description that best matched $\boldsymbol{x}$, thus accounting for linguistic expressivity within each relation. These best matches are then used to identify the relation expressed in $\boldsymbol{x}$. Mathematically:

$$y* = \arg\max_{i=1}^{|\mathcal{R}|} \; C\left(F\left(x*, d, \boldsymbol{W}\right);\; d \in \mathcal{D}^{r_i}\right) \tag{4.14}$$

where $\mathcal{R}$ is the set of relations present in the evaluation split, $\mathcal{D}^{r_i}$ is the set of relation descriptions of relation i : $i \in \{1, 2, \ldots, |\mathcal{R}|\}$, d is an actual relation description and $C\left(\cdot\right)$ is a function that composes the scores of all descriptions in the set $\mathcal{D}^{r_i}$. For our work we chose $C\left(\cdot\right)$ to be the arg max, but different functions, such as the average, could potentially be used, which is the reason why we present 4.14 in such a way.

When it comes to training time the truth is that the method above becomes significantly expensive, due to the sheer number of total relation descriptions that is considered. While it is manageable to make use of equation 4.14 for relatively small splits, such as the validation or test splits, it becomes prohibitively expensive to do so for all the data instances in a train split.

We approach this issue by, for each minibatch $\mathcal{M}$, sampling a relation description per relation, giving us $\mathcal{D}^{\mathcal{M}} = \bigcup_{i=1}^{|\mathcal{R}|} \{d^{r_i} \sim \mathcal{U}\left(\mathcal{D}^{r_i}\right)\}$, where $d^{r_i} \sim \mathcal{U}\left(\mathcal{D}^{r_i}\right)$ represents a description sampled uniformly from the set of descriptions associated with relation r_i. This way we have an order of magnitude fewer descriptions than at test time, making the whole training procedure significantly more tractable. Now we can simply compute[96] the score between each $\boldsymbol{x}_n \in \mathcal{M}$ and each $d \in \mathcal{D}^{\mathcal{M}}$, which we can turn into a categorical through the usage of the softmax function. Concretely, our

[96] *We leverage the fact that the computation of the score between $\boldsymbol{x}_n$ and any relation description d is entirely independent of the computation of the score between between $\boldsymbol{x}_n$ and any other relation description d'. Since the samples $\boldsymbol{x}$ are also assumed to be i.i.d, this allows us to parallelise all computations, by leveraging efficient tensor products between $\mathcal{M}$ and $\mathcal{D}^{\mathcal{M}}$.*

minibatch loss becomes, for a generic Loss function $L(\cdot)$:

$$\frac{1}{|\mathcal{M}|}\sum_{m=1}^{|\mathcal{M}|} L\left(y_m,\ \underset{i=1}{\overset{|\mathcal{R}|}{\arg\max}}\ \frac{\exp\left(F\left(x_m, d^{r_i} \in \mathcal{D}^{\mathcal{M}}; \boldsymbol{W}\right)\right)}{\sum_{d' \in \mathcal{D}^{\mathcal{M}}} \exp\left(F\left(x_m, d'; \boldsymbol{W}\right)\right)}\right) \tag{4.15}$$

The fact that we sample descriptions for each minibatch enables our models to implicitly learn that each relation has an underlying distribution of possible descriptions, which hopefully helps it generalise across different descriptions for both *seen* classes and *unseen* classes.

4.2 Evaluation Metrics

In this section we briefly explain the different metrics that we use to evaluate the performance of our models on the (G)ASL splits. We start with the Cross Entropy Loss in 4.2.1, followed by, in 4.2.2, the use of Accuracy as a metric and its pitfalls in imbalanced datasets. As a way to address the issue of evaluation of performance on imbalanced data, we discuss the alternative F1-Score (4.2.3), its multi-class variant, the Macro F1-Score (4.2.4) and finally introduce the Harmonic Macro F1-Score (4.2.5), as a more principled way of evaluating both (G)FSL train splits and GASL evaluation splits.

4.2.1 Cross Entropy Loss

Based on Information Theory concepts, the Cross Entropy Loss has a broad and common presence in ML classification problems. Abstractly, the Cross Entropy provides a measure of how similar two distributions, p and q, are.

In the specific case of ML classification, it compares the predicted class probabilities to the instance's actual class. Concretely, for a one-out-of-many classification problem with K classes, an instance x with label y and a model that estimates class probabilities $\boldsymbol{p} = (p_{c_1}, p_{c_2}, \ldots, p_{c_K})$, such that $\sum_{k=1}^{K} p_{c_k} = 1$, the Cross Entropy Loss is defined as:

$$\text{CE Loss} = \sum_{k=1}^{K} \delta_{y,k} \log\left(p_{c_k}\right) \tag{4.16}$$

where $\delta_{y,k}$ is the Kronecker delta[97].

[97] $\delta_{y,k} = 1$ *if* $y = k$
$\delta_{y,k} = 0$ *if* $y \neq k$

When taken for a minibatch $\mathcal{M} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$, the

result is computed independently and often averaged:

$$\text{CE Loss} = \frac{1}{|\mathcal{M}|} \sum_{m=1}^{|\mathcal{M}|} \sum_{k=1}^{K} \delta_{y_m,k} \log\left(p_{m,c_k}\right) \tag{4.17}$$

This computed value is then used during the backpropagation phase of the stochastic optimisation algorithm, giving rise to updated parameters.

In our specific case it is important to note that the classes that the probability distribution spans change from data split to data split. That is, if during training we optimise for a certain set of K classes, that does not necessarily mean that we evaluate (in either the validation or test splits) on the same classes or even just the same number of classes. Consequently, the underlying class distribution that was learned during training is entirely different from the one estimated at evaluation time, which, from a mathematical perspective, renders the evaluation time probability distribution as being improperly expressed.

Nonetheless, this is not an issue during training, as the classes that are considered do not change during the process, making it a viable approach to train a model. Still, at evaluation time, the Cross Entropy Loss can be used as an *indication* of how the model performs. On the other hand, at evaluation time we are only concerned with the highest scoring class, and not with the underlying probability distribution. This can be more easily assessed through the usage of the Accuracy metric, which shall now be discussed.

4.2.2 Accuracy

The Accuracy metric is very simple and we shall only discuss it very briefly. It basically consists of the percentage of correct classifications, for a certain set of instances. Specifically, for a set of data points, $\mathcal{P} = ((\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N))$, to be classified based on class set $\mathcal{C}$ and a model that predicts $y_n^* = \underset{y' \in \mathcal{C}}{\arg\max}\, F(x_n^*, y'; \boldsymbol{W})$ the Accuracy[98], Acc, is:

[98] *In per cent.*

$$Acc = 100 \cdot \frac{\sum_{n=1}^{N} \delta_{y_n, y_n^*}}{|\mathcal{P}|} \tag{4.18}$$

However, the Accuracy metric is not particularly suitable for imbalanced datasets. In particular, if model is biased to always predicting the more frequent classes, it will give the impression that it can perform extremely well, when the model might in fact be completely overlooking the less frequent classes. This aspect of the Accuracy metric brings us to the F1-Score, and its multi-class classification variants.

4.2.3 F1-Score

The F1-Score is a metric used to evaluate binary classification problems, where there is a significant data imbalance. It tries to address this imbalance by taking into account both the algorithms' precision and its recall. That is, it tries to balance how many correct predictions the algorithm makes, overall, but also how good it is at identifying all relevant elements. This can be defined as:

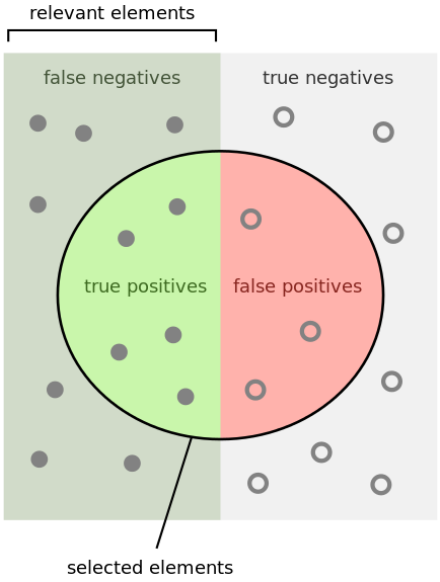


Figure 4.5: True/False Positives/Negatives

$$Precision = \frac{TP}{TP + FP} \tag{4.19}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.20}$$

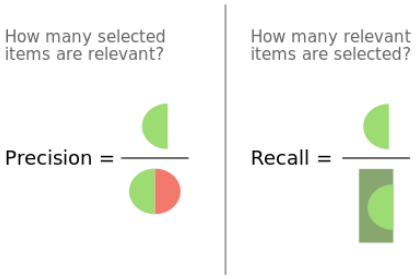


Figure 4.6: Precision/Recall

Images adapted from Wikipedia.
https://en.wikipedia.org/wiki/File:Precisionrecall.svg.

However, having the precision and recall values is not enough. Like we said, the F1-Score balances them. In particular, the F1-Score computes the Harmonic Mean[99] between the precision and the recall:

$$\text{F1} = 2 * \frac{Precision \cdot Recall}{Precision + Recall} \tag{4.21}$$

While more apt at dealing with data imbalance, the F1-Score is a binary classification metric. For our case we need a metric that can deal with more classes, which leads to the Macro F1-Score.

[99] *Other variants of the F1-Score exist, the general Fβ-Score, where more emphasis is placed on either precision or recall.*

4.2.4 Macro F1-Score

The Macro F1-Score takes a one-vs-all approach in order to extrapolate the F1-Score to a multi-class classification problem.

Simply stated, for a class set $\mathcal{C} = \{c_1, c_2, \ldots, c_K\}$, the Macro F1-Score computes the F1-Score for each individual class c_i, where it treats the elements of that class as positive labels, and the elements of any other class as negative labels[100].

[100] *Hence the one-vs-all approach.*

Furthermore, the Macro F1-Score places equal emphasis in all classes, by simply averaging the F1-Scores:

$$\text{MF1} = \frac{\text{F1}_{c_1} + \text{F1}_{c_2} + \cdots + \text{F1}_{c_K}}{K} \tag{4.22}$$

The Macro F1-Score can be, and is, used for (G)ASL evaluation settings. Yet, in some specific settings and particular splits, one needs to take into account both *seen* and *unseen* classes. With the Harmonic Macro F1-Score we extend the Macro F1-Score to better assess performance in these scenarios.

4.2.5 Harmonic Macro F1-Score

The principle behind (G)ASL is that we want to train a model that is able to extrapolate to previously *unseen* classes. More so, its Generalised counter part, GASL, aims at an evaluation that is also performed taking into account the classes seen during training time.

From the perspective of applications this is a far more realistic scenario. Accordingly, we want the model to perform well not only on the *unseen* classes but also the *seen* classes, as hardly any application would be able to know beforehand whether an instance belonged to one set or the other.

We argue on attempting to train models that perform equally well on both *seen* and *unseen* classes. As such, we propose that settings that contain instances of both *seen* and *unseen* classes should be evaluated with the Harmonic Mean between the individual Macro F1-Scores of each of the *seen*/*unseen* sets, thus placing equal emphasis on each of them:

$$\text{HMF1} = 2 * \frac{\text{MF1}_{seen} \cdot \text{MF1}_{unseen}}{\text{MF1}_{seen} + \text{MF1}_{unseen}} \tag{4.23}$$

Now that we have an appropriate set of metrics, with which to evaluate our models, we turn to the analysis of their performance on the proposed (G)ASL splits, along with a few extra evaluations.

4.3 Experiments

In this section we present the results from running both our models on the (G)ASL proposed splits. Additionally, we will also discuss some other performed experiments, along with their respective insights.

Concretely, in section 4.3.1 we discuss the particular implementation details that were used. In section 4.3.2 we briefly discuss some preliminary experiments that motivate our choice of models' structure. In section 4.3.3 we present our Relation Classification (RC) experiments on the proposed (G)ASL splits, which also allow us to confirm the more realistic inherent difficulty of the Generalised case. In section 4.3.4 we discuss experiments that are related to the unmasked and NER-masked versions of our splits. Finally, in section 4.3.5 we discuss our attempts at learning how to classify relations in an unsupervised way.

4.3.1 Implementation Details

Here we briefly discuss our choice of Deep Learning Framework, what kind of word embeddings we used, the properties used for training and the final configurations of our models, chosen through a hyperparameter grid search.

Deep learning Framework

Both our models, along with all the structure necessary to evaluate them, are built using the Deep Learning framework of PyTorch v1.0.0 (Paszke et al., 2017, [49]) and trained using Nvidia GTX 1080 Ti and GTX TITAN X GPUs. We make our code publicly available, under the MIT License, at `https://github.com/Nuno-Mota/MSc-AI---UvA---Master-Thesis`.

Word Embeddings

We make use of pretrained ELMo word embeddings (Peters et al., 2018, [51]), where we concatenate all three embedding types, contextual, deep and character-based, into a single 3072 sized float vector. The usage of ELMo embeddings, and in particular its character based part, allows us to not worry about which words are or are not part of the vocabulary, besides leveraging the quality of the word embeddings themselves.

For each data type, i.e. unmasked, subject/object masked and NER-masked, we precompute the word level ELMo embeddings for each unique sentence in the corresponding (G)ASL splits and dump them into a single[101] hdf5 file for easy storage and access. Computing the ELMo embeddings on the fly, that is, while running an experiment, would significantly increase the overall runtime and the ELMo model itself would take up critical GPU RAM memory.

[101] *If we created separate files for each setting/fold/split this would take too much storage space. With a single file this still requires around 300GB per data type.*

Stochastic Training Properties

For training we use the Adam optimiser (Kingma and Ba, 2014, [33]), with a learning rate of 0.001 and a batch size of 128.

We train for a maximum of 15 epochs when performing hyperparameter tuning (HT dataset) and for 25 epoch when performing our final evaluation (FE dataset). As mentioned before, we make use of early stopping determined on the validation set. If for 4 consecutive epochs the loss does not increase by more than 0.1 and the MF1, or HMF1[102], score does not increase by more than 0.5%, training is stopped. For evaluation on the test set we pick the model's state that performed the best on the validation set.

[102] *Depending on the setting being evaluated.*

BilSTM Baseline Hyperparameter Tuning

We perform hyperparameter selection for our BiLSTM baseline by evaluating its performance on the HT subject/object masking dataset. We selected the hyperparameter configuration that performed the best, when averaged over the different NL and (G)ASL settings.

For our baseline we tested the usage of two distinct BiLSTMs, one for $\boldsymbol{x}$ and another for $\boldsymbol{y}$, or simply one for both. As will be discussed in subsection 4.3.2, the single BiLSTM formulation proved to be better.

Additionally we tested our baseline's performance for an increasing LSTM hidden dimension size, specifically for sizes $50, 100, 200, 300, 400, 500$. A dimension of 500 proved to be the most beneficial in general, and performance seemed to start plateauing around size 400. Accordingly, a dimension of 500 and was thus chosen. Figure 4.8 illustrates the increase for a few of the (G)ASL settings.

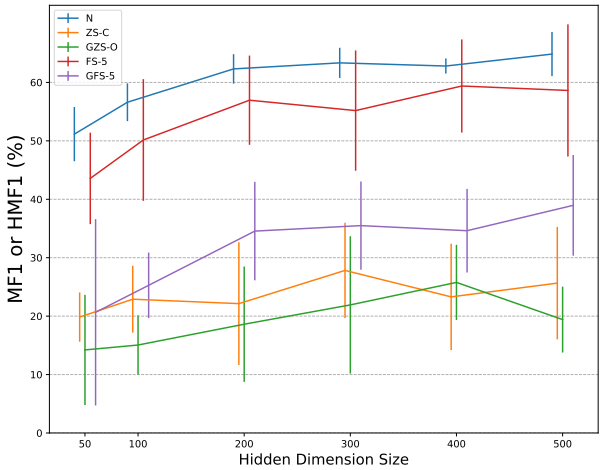


Figure 4.7: Performance Increase with Hidden Dimension Size

We randomly picked a few settings for exemplification. Specifically the NL setting, the ZSL setting (for the Closed Set framework), the GZSL setting (for the Open Set framework) and the (G)FSL-5 settings.

ESIM Hyperparameter Tuning

Like for our baseline, we perform hyperparameter selection for ESIM by evaluating its performance on the HT subject/object masking dataset. We selected the hyperparameter configuration that performed the best, when averaged over the different NL and (G)ASL settings.

Similarly, we also tested for the usage of two distinct BiLSTMs, one for $\boldsymbol{x}$ and another for $\boldsymbol{y}$, or simply one for both. As will be discussed in subsection 4.3.2, the single BiLSTM formulation proved to be better.

Since for ESIM we make use of word embeddings we also tested configurations that used a pretrained embedding layer. This will be discussed more in depth in subsection 4.3.2, but the conclusion is that there was no benefit from using the pre-trained layer. Accordingly, we do not pre-train the embedding layer for our final configuration.

As we mentioned in the description of ESIM (4.1.2), we replaced all ReLUs with LeakyReLUs. We also performed hyperparameter search for their negative slope, which we choose to be the same for all of them. We tested for values of 0.025 and 0.075. 0.075 turned out to be the best option.

For layers' sizes, we decide to assign the same size to all adjustable intermediate dimensions of ESIM, following the same approach as Chen et al. (2017, [11]) and Obamuyide and Vlachos (2018, [46]). Specifically, we tested for sizes of 50, 100 and 200. While we could have tested for larger layer sizes, this would significantly increase the memory requirements of our variant of ESIM and, as such, we chose not to. Out of the three possible values, models with intermediate layer sizes of 200 performed the best, as can be seen in the following figure[103], for a few of the (G)ASL settings:

[103] *All other hyperparameters are the ones corresponding to the best overall configuration.*

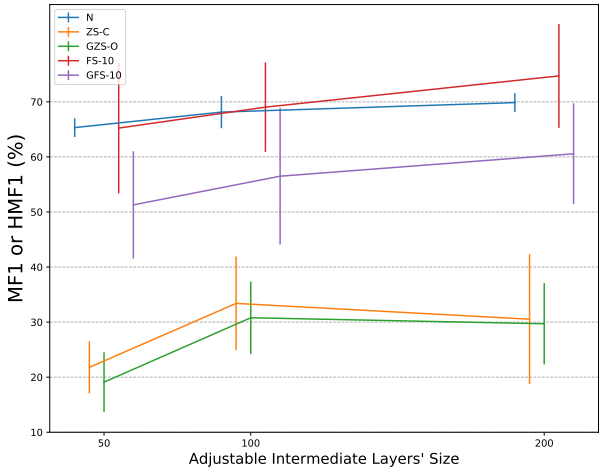


Figure 4.8: Performance Increase with Size of Intermediate Layers

We randomly picked a few settings for exemplification. Specifically the NL setting, the ZSL setting (for the Closed Set framework), the GZSL setting (for the Open Set framework) and the (G)FSL-10 settings.

4.3.2 Preliminary Experiments

In this subsection we discuss the benefits of using one or two BiLSTMs for both our Baseline and ESIM. We also cover the analysis of potentially using a pretrained embedding layer for ESIM.

One vs Two BiLSTMs - The Manifold Matching Problem

When we presented our models' architectures we proposed encoding sentences or words using one unique BiLSTM, as opposed to using the two distinct BiLSTMs that are common in NLI.

Our main intuition for this arose due to the ESIM's model attention mechanism being based on a basic inner product approach. This could be a problem, or at least hinder the models.

The problem itself happens due to each BiLSTM learning to project their respective data to a manifold that, theoretically, allows for a better distinction of the data's characteristics. However, the manifold of the BiLSTM that encodes $\boldsymbol{p}$ might not match the manifold of the BiLSTM that encodes $\boldsymbol{h}$, and for our particular case $\boldsymbol{x}$ and $\boldsymbol{y}$. What we mean by this is that even though $\boldsymbol{h^x}$ and $\boldsymbol{h^y}$ might carry the same underlying meaning, their vectorial representations are not necessarily similar, i.e. $\boldsymbol{h^x} \not\approx \boldsymbol{h^y}$.

Since an inner product yields a higher result (and consequently a higher attention weight) for similar vectors, it seems logical that we would want embeddings that encode similar words to actually have similar vectors. This is in fact the basic principle employed when producing word embeddings in an unsupervised way, such as with SkipGram (Mikolov et al., 2013, [39]).

Overall, the comparison of each embedding would then require either:

- **Option 1**: An extra component that would learn how to match the two manifolds.

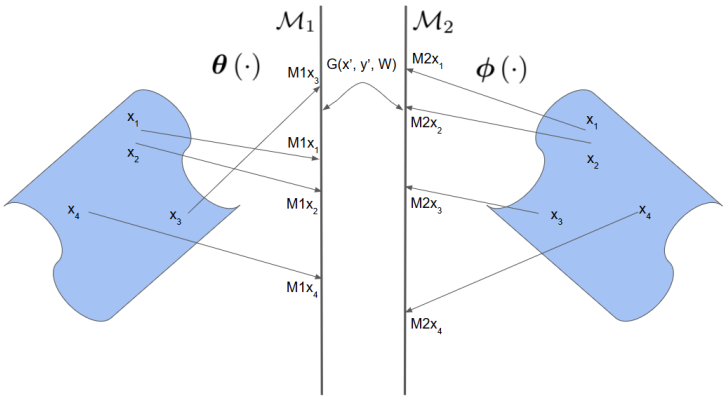


Figure 4.9: Diagram of Learning how to Match Different Manifolds

This is essentially what methods like Aligned Latent Embeddings (ALE) (Akata et al., 2016, [1]) are doing. However, they are matching different modalities (images/text) and, as such, there is not much else they can do. Neither do they rely solely on an inner product, meaning that even if they encode the same instances (let us assume for a moment that even, though they correspond to different modalities, each image/text instance could have a unique match) into non-matching manifolds, there is still a good chance that the function $G(\cdot)$ can learn how to match the two distinct manifolds.

- **Option 2**: Gradually match, during training, each of the manifolds.

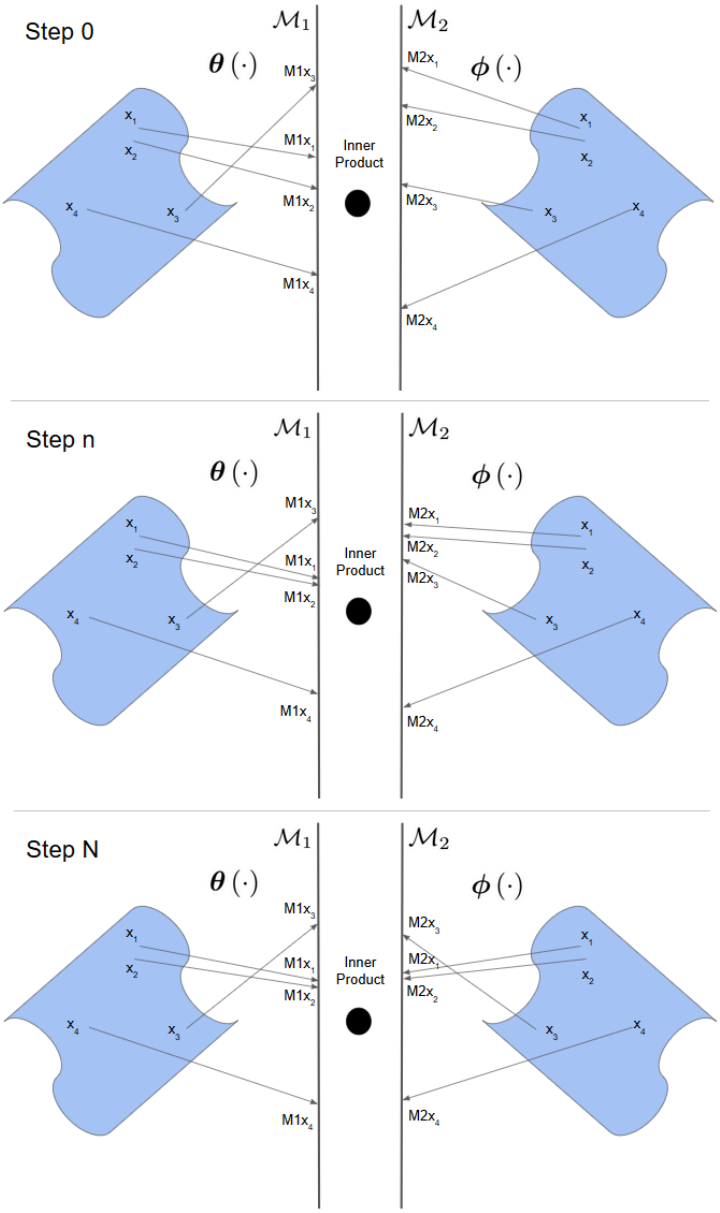


Figure 4.10: Diagram of Learning how to Directly Match Different Manifolds

Opposed to the previous option, here we do not have an intermediate learnable function. This means the the manifolds themselves must match, or else semantically equivalent instances might have different representations. This is the approach taken by methods like ESIM, even though they are working on the same data modality for each of the projections. This means that during training, their BiLSTMs must not only learn a meaningful separation of their own data, but also try to match it to the representation of the other BiLSTM.

- **Option 3**: Enforcing that the manifolds are the same, from the start.

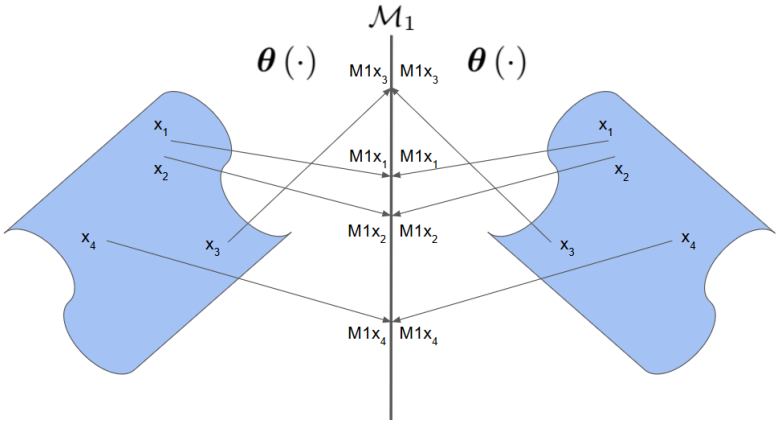


Figure 4.11: Diagram of Starting with Matched Manifolds

Option 3 allows us to completely avoid the problems presented by option 2, by simply using the same function to encode both embeddings (this assumes the data has the same modality, which is our case). In our particular case we ensure this by simply using the same BiLSTM to encode both $\boldsymbol{x}$ and $\boldsymbol{y}$. While it might slightly reduce the overall expressivity assigned to each of the sentences, individually, it ensures that their vectorial representation matches their underlying meaning, for both $\boldsymbol{x}$ and $\boldsymbol{y}$.

Regardless of our convictions, we looked into the last two options, as they are more in line with the initial components of our more complex model (ESIM).

In order to test whether using a single BiLSTM is better than using two distinct ones, we made use of out HT dataset and simply treated this aspect as an hyperparameter of both our Baseline and our ESIM model.

Furthermore, we then computed, for each fixed hyperparameter configuration (leaving the single or dual BiLSTMs hyperparameter unfixed) and each setting (NL, ZSL, ...), the difference between the performance of the single BiLSTM and the dual BiLSTM configuration. Specifically, we subtract the achieved score (which can be either the MF1 metric or the HMF1

metric, depending on the setting) of the dual configuration to the single configuration: metric_diff(single BiLSTM, dual BiLSTMs). Thus, a positive difference means that the single BiLSTM configuration performed better than the dual (and, if the difference is negative, the other way around).

Finally, we average over the folds of the specific setting and determine significance with a two-tailed t-test with a p-value of 0.05. The reason that we evaluate each pair (configuration, setting) individually is that the t-test assumes that (or rather, is stronger when) the two populations have the same controlled variables, besides the one being tested for.

Table 4.1 illustrates this for the hyperparameters of our Baseline and the (G)ZSL settings (where we also evaluate the Open and Closed Set Frameworks). For our baseline we make available the overall results in A.3[104].

[104] *We do not do so for ESIM due to the size that the tables would take.*

Hidden Layer Size	ZS		GZS	
	Open-Set	Closed-Set	Open-Set	Closed-Set
50	6.55 ± 6.69	**8.47 ± 3.04**	5.10 ± 6.13	1.45 ± 1.99
100	6.30 ± 6.54	**10.47 ± 4.38**	3.54 ± 5.79	3.05 ± 4.20
200	5.62 ± 7.05	5.00 ± 8.04	6.24 ± 14.36	1.18 ± 1.73
300	7.87 ± 10.19	**13.22 ± 7.88**	**15.55 ± 9.97**	0.61 ± 0.94
400	10.36 ± 5.82	3.09 ± 10.10	**14.86 ± 4.81**	1.54 ± 4.32
500	8.70 ± 13.24	8.27 ± 10.44	5.08 ± 2.89	-1.69 ± 3.71

Table 4.1: Difference of using 1 vs 2 BiLSTM(s) (Baseline - (G)ZSL settings)

This table shows, for our baseline and the (G)ZSL settings, the differences of using a single BiLSTM vs using two BiLSTMs, i.e. we compute metric_diff(1 BiLSTM, 2 BiLSTMs) for each HT fold and then average over folds. All other hyperparameters are fixed. Bold face results indicate statistical significance for the difference not being 0, at a p-value of 0.05, under a t-test.

What can we conclude from our results? One thing we can do is observe certain trends that indicate which formulation is better overall. For example, we can analyse how many of our pairs favour one configuration or the other and, how many of those represent statistically significant results. We present this information in the table below, for both our models:

		Difference in favour of a single BiLSTM			
Model	# pairs	+	Signif. +	-	Signif. -
Baseline	78	63	9	15	0
ESIM_StS	156	149	60	7	1

Table 4.2: Differences in Favour of a Single BiLSTM

From this information it is very clear that both models perform better when using a single BiLSTM. In particular, ESIM seems to benefit the most, with around 40% overall significant cases favouring it, not to mention that 95% of all cases, significant or not, seem to benefit from the single BiLSTM configuration. While we cannot argue that those cases have a

statistical meaning by themselves, when taken together we can admit the possibility of them being relevant as a whole. Regardless, when faced with 60 significant cases in favour of a single BiLSTM versus a single one in favour of the dual configuration, the choice becomes easy.

Interestingly enough, the results are less stark when it comes to our Baseline. We believe this might be due to the fact that it is encoding the entire sentences as a whole. This probably means that it almost only needs to capture information directly regarding the classes, as opposed to ESIM, which uses this BiLSTMs to encode the meaning of individual words. These are in far greater number and are also necessary to perform the attention stage, making it more important to accurately match each of the manifolds. With the single BiLSTM we can ensure that each hidden representation has the same vector, as desired.

Pretrained Embedding Layer (ESIM's Input Encoding)

Since we make use of pretrained ELMo embeddings (Peters et al., 2018, [51]), our ESIM input encoding component can be seen instead as a dimensionality reduction method, that hopefully still preserves the semantic information, or most of it, present in the original ELMo embeddings.

We investigated if it was possible to pretrain this part of ESIM, as that would mean that at runtime the model would not have to learn the projection from scratch, which would potentially speed up the training procedure. More importantly, we can pretrain it using all the sentences in our (G)ASL splits, not only the ones present in the train split which is used to train the model. This means that our input encoding BiLSTM could potentially learn more meaningful representations.

In order to achieve this pretraining we build an Auto-Encoder (AE) (Hinton and Salakhutdinov, 2006, [29]). Our encoder is the BiLSTM present in the input encoding component of ESIM and our decoder is another BiLSTM that projects the hidden word embeddings, produced by the encoder, back to their original size. That is, if our encoder BiLSTM is a function f that, for each word, maps[105] from $f : \mathbb{R}^{3072} \rightarrow \mathbb{R}^{d_h}$, where d_h is the BiLSTM's hidden dimension, then our decoder BiLSTM, which we can represent as function g, is a function that maps from $g : \mathbb{R}^{d_h} \rightarrow \mathbb{R}^{3072}$.

As all the information we require is contained in the set of all sentences present in the (G)ASL splits, we do not need our model to be able to generalise to unseen input sentences. Subsequently, we choose to train our AE by only using a train set, which consists of all 996.361 unique sentences of the (G)ASL splits, for a max of 5 epochs[106].

[105] *Note that we assume our input to be the concatenated ELMo embeddings, which have a dimension of 3072.*

[106] *Since we are considering close to 1.000.000 instances training instances.*

In order to evaluate the benefit of using such pretrained configurations, we follow a procedure similar to the one used in our Manifold Matching analysis. Concretely, we fix all other hyperparameters[107] and pair their unique configurations with each of the HT dataset settings. Then, for each pair, we subtract the score[108] of the pretrained version to that of the non-pretrained version: metric_diff(non-pretrained, pretrained). Thus, a positive difference means that the non-pretrained version performed better than the pretrained one (and if the difference is negative we have the other way around). Finally, we average over the folds of the specific setting and determine significance with a t-test with a p-value of 0.05. Like in the Manifold Matching problem, we analyse our results by assessing how many (configuration, setting) pairs indicate that the usage of one formulation is better than the other. The results are as follow, for the 156 possible (configuration, setting) pairs:

[107] *That is, all except for the usage or not of a pretrained input encoding BiLSTM.*

[108] *Which can be the MF1 or the HMF1 metric, depending on the setting.*

+	Signif. +	-	Signif. -
88	2	68	3

Table 4.3: Differences in Favour of Pretraining ESIM's Input Encoding Component.

Surprisingly, our results seem to indicate that for ESIM there is no difference between using a pretrained input encoding BiLSTM or simply learning it from scratch.

We suspect that this just means that the model is capable of learning the necessary representations by itself rather quickly, thus rendering the pretraining, of the input encoding BiLSTM, nearly irrelevant.

Now that we have defined the configuration for both our models, we shall evaluate their performance on the (G)ASL splits of the FE dataset.

4.3.3 (G)ASL Relation Classification

For evaluating how both our models perform on the (G)ASL splits, we will first focus on the Normal Learning (NL) setting, followed by an analysis of the Any-Shot Learning (ASL) settings. Finally, we move to their Generalised counter part, the Generalised Any-Shot Learning (GASL) settings, where we also compare the performance on the *unseen* classes in the Generalised case against the non-Generalised one.

Normal Learning Setting

In the Normal Learning (NL) setting all 120 existing classes are present in all three splits (train, validation and test). While this might give the model a broader set of features with which to train, we also followed a strict data distribution when creating the splits. As such, it is quite possible that

some classes will have had very few instances at training time. This can impact the performance of our models according to the MF1 metric, as it takes into account the performance on all classes equally. Even so, it should affect both models in the same fashion, which means that any difference in performance will be due to a better generalization capability.

Below we illustrate the evolution of the performance of each of our models on the validation splits:

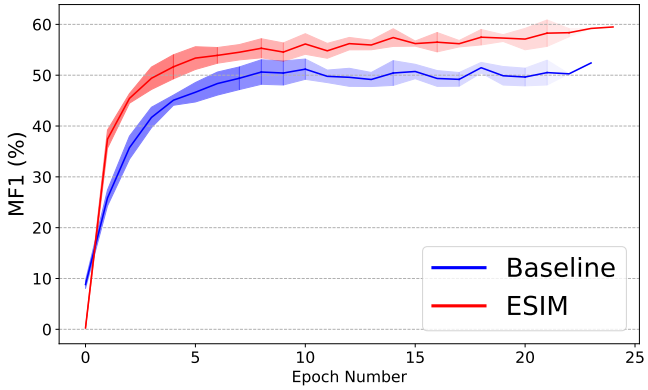


Figure 4.12: NL Setting - Performance on the Validation Splits

Evolution, for both models, of the performance on the validation splits of the NL setting. The solid line indicates the average over the folds of the NL setting. The shaded area indicates the corresponding standard deviation. The intensity of the shaded area is proportional to the number of folds still being evaluated at the corresponding epoch number (which might decrease, due to early stopping).

Surprisingly, such a simple model as our BiLSTM baseline can still perform reasonably well, particularly when we consider the data's long tail distribution present in the training split. As for ESIM, we can see that it's capable of picking up patterns in the data more quickly than our simple Baseline and maintains an overall advantage throughout the training procedure.

Furthermore, ESIM's greater performance is also verified at test time, with a statistically significant advantage, as can be seen in the table 4.4:

Model	NL
Baseline	54.92 ± 1.15
ESIM	**59.05 ± 2.52**

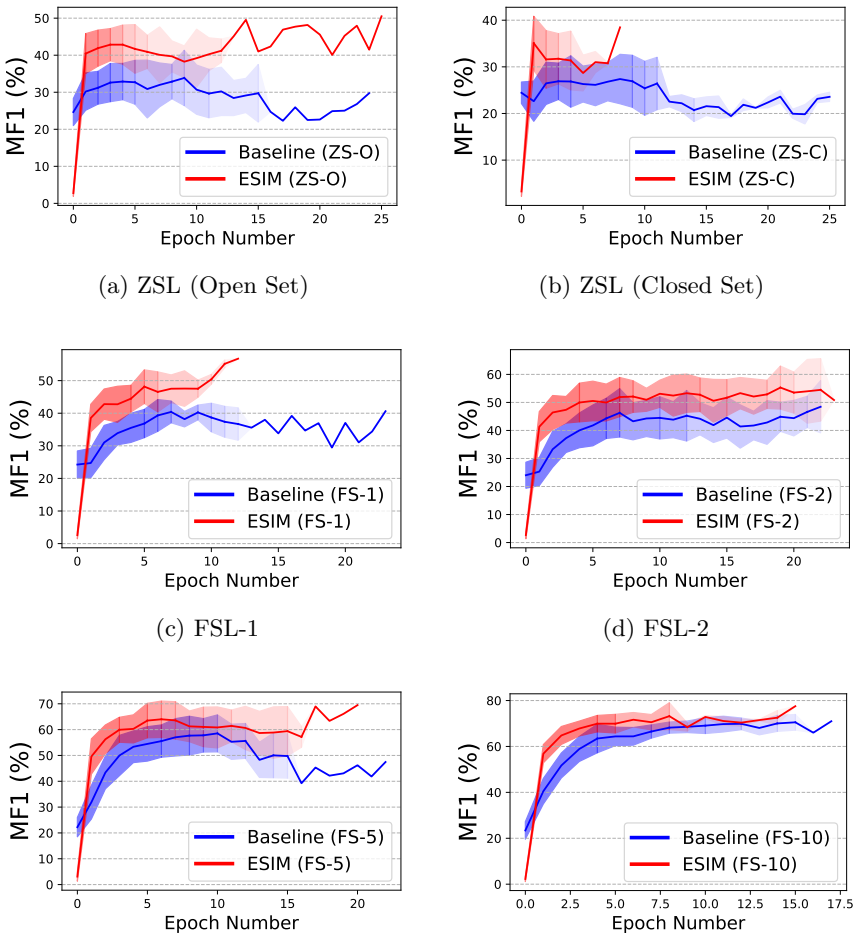
Table 4.4: FE - Comparison of Baseline and ESIM for the NL Setting

This table shows, for the FE dataset and the NL setting, the MF1 test scores, in %, for both our baseline and our ESIM variant (ESIM_StS). Bold face results indicate the best result, with two-tailed t-test statistical significance, at a p-value of 0.05.

Any-Shot Learning Settings

We now discuss our models' performance on the Any-Shot Learning (ASL) settings. Concretely, these are the settings that evaluate only on the *unseen* classes and that at training time have 0 instances (ZSL), 1 instance (FSL-1), 2 instances (FSL-2), 5 instances (FSL-5) or 10 instances (FSL-10) of the evaluation classes.

As we mentioned before, the comparison of a ZSL setting with FSL settings would be more just if the ZSL setting was trained under the Closed Set assumption. As such, we train our models on both a ZSL Open Set Framework and a Closed Set one. Below, we provide a comparison of the evolution of the performance of our models, for all ASL settings:



(a) ZSL (Open Set)

(b) ZSL (Closed Set)

(c) FSL-1

(d) FSL-2

(e) FSL-5

(f) FSL-10

Figure 4.13: ASL Settings - Performance on the Validation Splits

Evolution, for both models, of the performance on the validation splits of the ASL settings. The solid line indicates the average over the folds of the respective setting. The shaded area indicates the corresponding standard deviation. The intensity of the shaded area is proportional to the number of folds still being evaluated at the corresponding epoch number (which might decrease, due to early stopping).

One thing that stands out is that during training our more complex ESIM model seems to perform better on all settings. However, there's quite an overlap in both models' uncertainty estimation, which might mean that the results are not statistically significant.

We resort to the test set to be able to draw more precise conclusions:

Model	ZSL-O	ZSL-C
Baseline	32.90 ± 3.87	23.90 ± 5.46
ESIM	**39.35 ± 4.87**	**32.81 ± 5.51**

(a) ZSL Open and Closed Settings

	FSL			
Model	1	2	5	10
Baseline	41.74 ± 4.48	46.39 ± 4.10	59.06 ± 7.42	67.31 ± 5.36
ESIM	44.01 ± 6.48	**52.20 ± 6.76**	63.54 ± 7.13	71.73 ± 4.42

(b) FSL Settings

Table 4.5: FE - Comparison of Baseline and ESIM for the ASL Settings

This table shows, for the FE dataset and the ASL settings, the MF1 test scores, in %, for both our baseline and our ESIM variant (ESIM_StS). Bold face results indicate the best result, between both models, with two-tailed t-test statistical significance, at a p-value of 0.05.

In fact, ESIM only performs better, with a statistical significance, in three of the six overall settings. Nonetheless, it never performs worse, than our Baseline. This seems to indicate that, in general, ESIM benefits from the added complexity when tackling ASL settings. Moreover, it is surprising that a model as simple as our Baseline can generalise so well and also learn so much information from such a limited number of instances.

We now turn towards the comparison of the ZSL Open and Closed Set Frameworks. Do the models really perform significantly worse when they constantly receive a negative learning signal for the *unseen* classes[109]? Again, we test for statistical significance when comparing one framework to the other. Table 4.6 provides results:

[109] *That is, when evaluated under the Closed Set, as opposed to the Open one.*

	Model	
Framework	Baseline	ESIM
Open Set	**32.90 ± 3.87**	**39.35 ± 4.87**
Closed Set	23.90 ± 5.46	32.81 ± 5.51

Table 4.6: FE - Comparison of ZSL Open and Closed Frameworks

This table shows, for the FE dataset, the MF1 test scores, in %, for both our baseline and our ESIM variant (ESIM_StS) when comparing the ZSL Open Set Framework to the ZSL Closed Set Framework. Bold face results indicate the best result, between both frameworks, with two-tailed t-test statistical significance, at a p-value of 0.05.

In this case the results are clear, unsurprisingly, one might say. The models benefit significantly from not considering the *unseen* classes at training time, seeing as they can never push their weights towards recognizing the *unseen* classes' features, as these classes have no instances whatsoever. As such, if one does not use more involved (G)ASL methods[110], it is better to train a ZSL model without even considering classes for which no training instances are available. That is, it is better to train a model for recognition.

[110] *Which we will briefly discuss when presenting possible Future Work.*

Finally we turn towards the study of the impact of increasing the number of instances available at training time, i.e. increasing the Shot Number of the ASL settings. We compare, for each model, individually, if there are significant improvements from one Shot number to the next one, in an increasing fashion. For a fair comparison, for the ZSL setting we only consider the Closed Set Framework. Table 4.7 provides the results and figure 4.14 provides a visualization:

	Model	
Setting	Baseline	ESIM
ZSL-C	23.90 ± 5.46	32.81 ± 5.51
FSL-1	**41.74 ± 4.48**	**44.01 ± 6.48**
FSL-2	**46.39 ± 4.10**	**52.20 ± 6.76**
FSL-5	**59.06 ± 7.42**	**63.54 ± 7.13**
FSL-10	**67.31 ± 5.36**	**71.73 ± 4.42**

Table 4.7: FE - Increasing the Shot Number in ASL Settings

This table shows, for the FE dataset, the MF1 test scores, in %, for both our baseline and our ESIM variant (ESIM_StS) when increasing the Shot number in ASL Settings. Bold face results indicate the best result, between 2 consecutive and increasing Shot numbers (e.g.: if FS-2 has statistically better results than FS-1, then FS-2 will be bold faced), with two-tailed t-test statistical significance, at a p-value of 0.05.

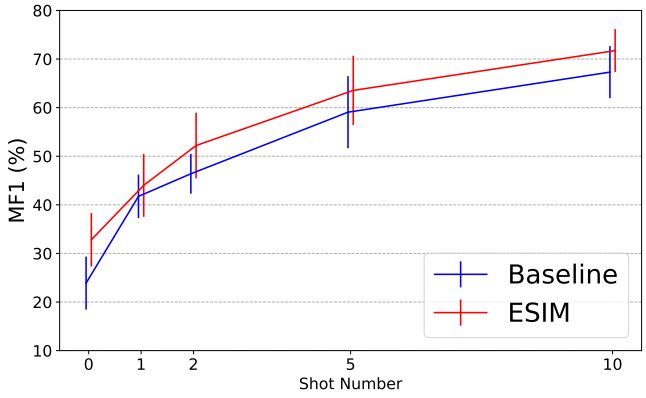


Figure 4.14: ASL Settings - Increasing the Shot Number

Visualization of the impact of increasing the Shot number, in ASL settings, for both our Baseline and ESIM.

As can be observed, both models definitely make use of the additional

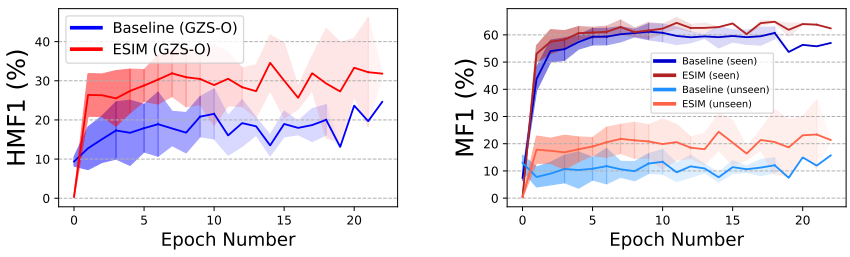
instances. In fact, both of them always achieve statistically significant better results between consecutively increasing Shot numbers. This shows that the presence of even a few instances, of the *unseen* classes, makes a profound difference in both models' generalisation capabilities.

Now that it is clear how both our models perform in ASL tasks, we will turn our attention towards the task of GASL.

Generalised Any-Shot Learning Settings

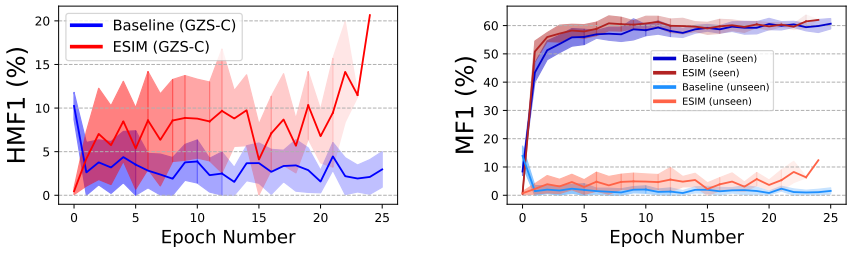
Generalised Any-Shot Learning (GASL) settings should, theoretically, provide a harder, yet more realistic, evaluation scenario, when compared to ASL settings, due to the fact that at evaluation time the *seen* classes are also considered, unlike what happens with the ASL settings. That is what we aim to show in this part of the discussion, as a motivation for future research to evaluate (G)ASL tasks uniquely under a Generalised case framework, i.e. on GASL tasks. Accordingly, we consider the performance of our models on the GZSL (for both the Open and Closed Set Frameworks), GFSL-1, GFSL-2, 5 GFSL-5 and GFSL-10 settings.

As we did with ASL, below we provide a comparison of the evolution of the performance of our models, for all GASL settings. Please note that, since we also include the performance on *seen*/*unseen* classes, which increases the overall number of figures, we split the figures into the corresponding GZSL and GFSL parts. The description for the GZSL figure, 4.15, is the same as the description of the GFSL figure, 4.16:



(a) GZSL (Open Set)

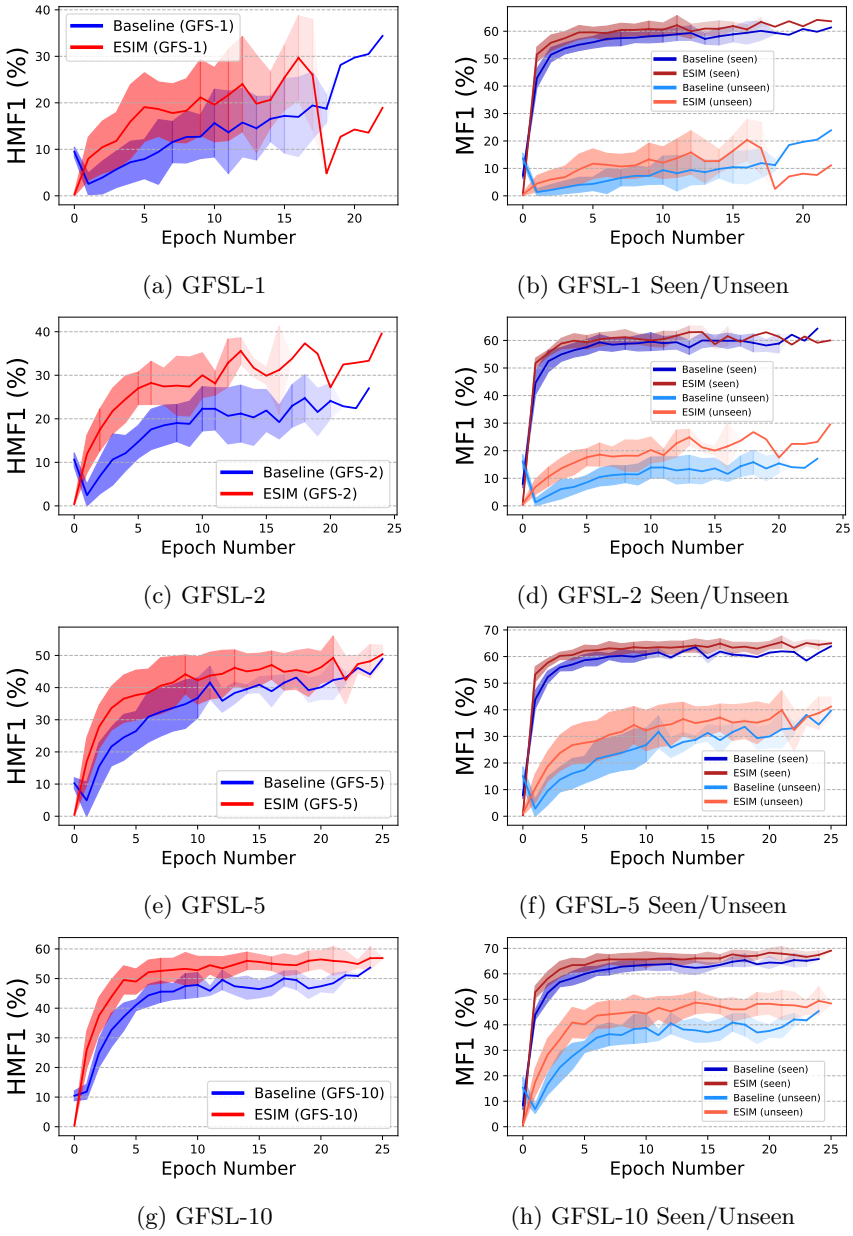
(b) GZSL (Open Set) Seen/Unseen



(c) GZSL (Closed Set)

(d) GZSL (Closed Set) Seen/Unseen

Figure 4.15: GZSL Settings - Performance on the Validation Splits



(a) GFSL-1

(b) GFSL-1 Seen/Unseen

(c) GFSL-2

(d) GFSL-2 Seen/Unseen

(e) GFSL-5

(f) GFSL-5 Seen/Unseen

(g) GFSL-10

(h) GFSL-10 Seen/Unseen

Figure 4.16: GFSL Settings - Performance on the Validation Splits

Evolution, for both models, of the performance on the validation splits of the GASL settings. The left column illustrates the overall models' performance according to the HMF1 metric. The right column illustrates the models' performance on each of the *seen*/*unseen* sets, separately. The solid lines indicate the average over the folds of the respective setting. The shaded areas indicate the corresponding standard deviation. The intensity of the shaded area is proportional to the number of folds still being evaluated at the corresponding epoch number (which might decrease, due to early stopping).

Similarly to the ASL case, the ESIM model appears to outperform our BiLSTM baseline, in both the overall score computed with the HMF1 metric but also on each individual *seen*/*unseen* set. Yet, there is also quite a big standard deviation for both models. We recourse once more to a statistical

significance analysis, which we split into the overall performance using the HMF1 metric and the performance on the individual *seen*/*unseen* sets:

	Seen & *Unseen* Classes (HMF1)	
Model	ZSL-O	ZSL-C
Baseline	18.23 ± 5.40	2.06 ± 2.11
ESIM	**28.54 ± 5.50**	**9.54 ± 4.29**

(a) GZSL Open and Closed Settings

	Seen Classes (MF1)	
Model	GZSL-O	GZSL-C
Baseline	57.72 ± 1.80	54.57 ± 2.14
ESIM	58.36 ± 2.00	**56.81 ± 1.93**

(b) GZSL Open and Closed Settings (*Seen* Classes) Only

	Unseen Classes (MF1)	
Model	GZSL-O	GZSL-C
Baseline	11.03 ± 3.99	1.07 ± 1.12
ESIM	**19.15 ± 4.81**	**5.30 ± 2.53**

(c) GZSL Open and Closed Settings (*Unseen* Classes) Only

	Seen & *Unseen* Classes (HMF1) GFSL			
Model	1	2	5	10
Baseline	13.58 ± 4.12	23.29 ± 3.69	34.80 ± 6.52	48.13 ± 4.65
ESIM	**22.74 ± 5.86**	**29.84 ± 5.16**	**45.64 ± 5.47**	**56.73 ± 3.74**

(d) GFSL Settings

	Seen Classes (MF1) GFSL			
Model	1	2	5	10
Baseline	55.32 ± 1.99	56.06 ± 2.64	57.17 ± 3.03	59.32 ± 2.52
ESIM	**58.04 ± 1.94**	57.91 ± 2.24	**61.24 ± 2.36**	**63.84 ± 2.60**

(e) GFSL Settings (*Seen* Classes) Only

	Unseen Classes (MF1) GFSL			
Model	1	2	5	10
Baseline	7.85 ± 2.63	14.78 ± 2.95	25.41 ± 6.56	40.68 ± 5.64
ESIM	**14.37 ± 4.46**	**20.30 ± 4.43**	**36.78 ± 6.73**	**51.23 ± 5.18**

(f) GFSL Settings (*Seen* Classes) Only

Table 4.8: FE - Comparison of Baseline and ESIM for the GASL Settings

This table shows, for the FE dataset and the GASL settings, the HMF1, or MF1 (for the exclusive *seen*/*unseen* cases), test scores, in %, for both our baseline and our ESIM variant (ESIM_StS). Bold face results indicate the best result, between both models, with two-tailed t-test statistical significance, at a p-value of 0.05.

The results are clear. In the case of the GASL task, ESIM performs significantly better, than our BiLSTM baseline, in all tasks. Even though

there are a couple of cases where ESIM does not perform significantly better on the *seen* classes, it does so on the *unseen* classes, over all settings.

This in turn influences the overall HMF1 based score. This evidence also allows us to advocate for the HMF1 as a suitable metric to evaluate performance on the Generalised case, particularly so when one is interested in achieving a good performance on both *seen* and *unseen* classes. As can be observed above, if the performance on either the *seen* or the *unseen* classes[111] is worse than the other *seen*/*unseen* set, then the overall HMF1 based score will be drastically reduced, clearly indicating that the model does not perform what is desired.

[111] *In our case it is always on the unseen classes.*

Also interesting, but unsurprising, is that we can clearly observe that the models' performance on the *seen* set does not vary that much across different Shot numbers. Yet, it does seem to increase slightly, suggesting it also makes use of the *unseen* classes features when identifying *seen* classes.

We now address the study of the difference in peformance between the GZSL Open Set Framework and the GZSL Closed Set Frameworks. In particular we focus on how it affects the overall HMF1 based performance and the more specific MF1 based performance on the *seen*/*unseen* classes.

	Seen & *Unseen* Classes (HMF1) Model	
Framework	Baseline	ESIM
Open Set	**18.23 ± 5.40**	**28.54 ± 5.50**
Closed Set	2.06 ± 2.11	9.54 ± 4.29

(a) Comparison on both *Seen* and *Unseen* Classes

	Seen Classes (MF1) Model	
Framework	Baseline	ESIM
Open Set	**57.72 ± 1.80**	58.36 ± 2.00
Closed Set	54.57 ± 2.14	56.81 ± 1.93

(b) GZSL Open and Closed Settings (*Seen* Classes) Only

	Unseen Classes (MF1) Model	
Framework	Baseline	ESIM
Open Set	**11.03 ± 3.99**	**19.15 ± 4.81**
Closed Set	1.07 ± 1.12	5.30 ± 2.53

(c) GZSL Open and Closed Settings (*Unseen* Classes) Only

Table 4.9: FE - Comparison of GZSL Open and Closed Frameworks

This table shows, for the FE dataset, the HMF1, or MF1 (for the exclusive *seen*/*unseen* cases), test scores, in %, for both our baseline and our ESIM variant (ESIM_StS) when comparing the GZSL Open Set Framework to the GZSL Closed Set Framework. Bold face results indicate the best result, between both frameworks, with two-tailed t-test statistical significance, at a p-value of 0.05.

Again, just like the ZSL Open/Closed Set comparison, the results are clear. We refer to that discussion, for the sake of brevity. Nonetheless, It is interesting to observe that, when evaluating *seen* classes, our Baseline also benefits from being trained in the Open Set framework. This could be due to the fact that during training the model does not need to assign any probability mass to the *unseen* classes, however little this might be. Therefore, it can "focus" on simply distinguishing between the *seen* classes, which in turn provides it a better capability to, at test time, identify them better.

Let us now shift our attention to the impact, on model performance, of increasing the Shot number. Like we did in the study of the ASL counterpart, we compare, for each model individually, if there are significant improvements from one Shot number to the next one, in an increasing fashion. Since we are now working in the Generalised case, we do this for the overall HMF1 based score and for the individual MF1 based score on both the *seen* and *unseen* classes. For a fair comparison, for the GZSL setting we only consider the Closed Set Framework. Table 4.10 provides the results and figure 4.17 provides a visualization:

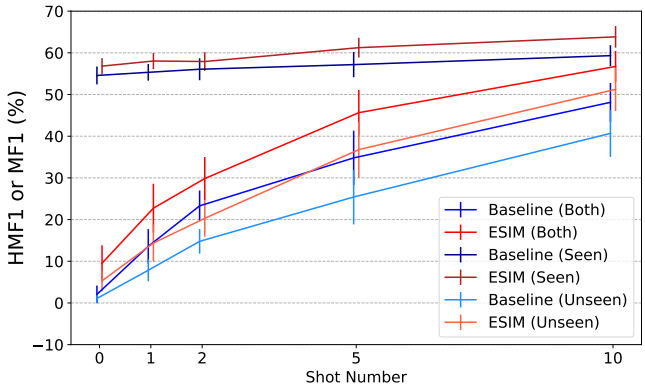


Figure 4.17: GASL Settings - Increasing the Shot Number

Visualization of the impact of increasing the Shot number, in GASL settings, for both our Baseline and ESIM.

Again, just like in the ASL task, it is clear that both models make use of the additional instances. In fact, both of them always achieve statistically significant better results, for both the overall HMF1 based performance and the MF1 performance on the *unseen* classes, between consecutively increasing Shot numbers. This shows that the presence of even a few instances, of the *unseen* classes, makes a profound difference in both models' generalisation capabilities. More interestingly, the addition of even a few instances of the *unseen* classes appears to also improve the models' performance on the *seen* classes, even if only statistically significant for a couple of cases. This is most likely due to extra information that the Few-Shot instances bring

	Seen & *Unseen* Classes (HMF1) Model	
Setting	Baseline	ESIM
GZSL-C	2.06 ± 2.11	9.54 ± 4.29
GFSL-1	**13.58 ± 4.12**	**22.74 ± 5.86**
GFSL-2	**23.29 ± 3.69**	**29.84 ± 5.16**
GFSL-5	**34.80 ± 6.52**	**45.64 ± 5.47**
GFSL-10	**48.13 ± 4.65**	**56.73 ± 3.74**

(a) Comparison of Increasing Shot Number, on both *Seen* and *Unseen* Classes

	Seen Classes (MF1) Model	
Setting	Baseline	ESIM
GZSL-C	54.57 ± 2.14	56.81 ± 1.93
GFSL-1	55.32 ± 1.99	58.04 ± 1.94
GFSL-2	56.06 ± 2.64	57.91 ± 2.24
GFSL-5	57.17 ± 3.03	**61.24 ± 2.36**
GFSL-10	59.32 ± 2.52	**63.84 ± 2.60**

(b) Comparison of Increasing Shot Number, on *Seen* Classes Only

	Unseen Classes (MF1) Model	
Setting	Baseline	ESIM
GZS-C	1.07 ± 1.12	5.30 ± 2.53
GFSL-1	**7.85 ± 2.63**	**14.37 ± 4.46**
GFSL-2	**14.78 ± 2.95**	**20.30 ± 4.43**
GFSL-5	**25.41 ± 6.56**	**36.78 ± 6.73**
GFSL-10	**40.68 ± 5.64**	**51.23 ± 5.18**

(c) Comparison of Increasing Shot Number, on *Unseen* Classes Only

Table 4.10: FE - Increasing the Shot Number in GASL Settings

This table shows, for the FE dataset, the HMF1, or MF1 (for the exclusive *seen*/*unseen* cases), test scores, in %, for both our baseline and our ESIM variant (ESIM_StS) when increasing the Shot number in GASL Settings. Bold face results indicate the best result, between 2 consecutive and increasing Shot numbers (e.g.: if GFS-2 has statistically better results than GFS-1, then GFS-2 will be bold faced), with two-tailed t-test statistical significance, at a p-value of 0.05.

during training. This information allows the model to have a better understanding of general relation features, and thus a likely better understanding of also the *seen* classes. Likewise, it probably helps in better distinguishing when an instance belongs to either the *seen* or *unseen* set, thus also improving the performance on the *seen* classes.

Finally, on table 4.11, we compare the models' performance on the *unseen* classes between the ASL settings and the GASL settings, as a way of highlighting the difference from the non-Generalised case to the Generalised case.

	Model			
	Baseline		ESIM	
Setting	Non-Generalised	Generalised	Non-Generalised	Generalised
(G)ZSL-O	**32.90 ± 3.87**	11.03 ± 3.99	**39.35 ± 4.87**	19.15 ± 4.81
(G)ZSL-C	**23.90 ± 5.46**	1.07 ± 1.12	**32.81 ± 5.51**	5.30 ± 2.53
(G)FSL-1	**41.74 ± 4.48**	7.85 ± 2.63	**44.01 ± 6.48**	14.37 ± 4.46
(G)FSL-2	**46.39 ± 4.10**	14.78 ± 2.95	**52.20 ± 6.76**	20.30 ± 4.43
(G)FSL-5	**59.06 ± 7.42**	25.41 ± 6.56	**63.54 ± 7.13**	36.78 ± 6.73
(G)FSL-10	**67.31 ± 5.36**	40.68 ± 5.64	**71.73 ± 4.42**	51.23 ± 5.18

Table 4.11: FE - Comparing Performance on the *unseen* Classes Between ASL and GASL

This table shows, for the FE dataset, the comparison, between the ASL and the GASL settings, of the MF1 test scores, in %, of, exclusively, the *unseen* classes, for both our Baseline and our ESIM variant (ESIM_StS). Bold face results indicate the best result, between the Generalised and Non-Generalised cases (independently of the model), with two-tailed t-test statistical significance, at a p-value of 0.05.

The results are no surprise, particularly so after the analysis that has already been performed, but they serve the purpose of demonstrating how misleading the ASL settings can be, when compared to the more realistic GASL settings.

In conclusion, the analysis that has been performed so far allowed us to:

- Compare both our Baseline and the more complex ESIM model and to conclude that the extra complexity present in the ESIM model is of significant importance, particularly so when evaluating on the Generalised case.
- Verify that, for model performance on the (G)ZSL settings, the Open Set Framework is indeed more beneficial than the Closed Set One.
- Verify the improvement in model performance when increasing the Shot number.
- Verify that good results on the ASL settings do not properly reflect the performance that a model would have in a more realistic scenario (such as the GASL settings).

Thus we conclude our analysis of the performance of both our models on the (G)ASL settings. Now, we shall briefly evaluate how they perform when no specific annotation is present, by evaluating their performance on our unmasked and NER-masked dataset counterparts, as discussed in chapter 3.

4.3.4 Importance of Masking

For the sake of simplicity, we assumed that the hyperparameters found for our previous (G)ASL RC task were also valid for both the Unmasked and NER-masked counterparts, though, in truth, they are different tasks that would benefit from their own hyperparameter tuning procedure.

As we pointed out earlier, the main point of this experiment was to understand how capable the models were of identifying the underlying relations when no specific annotation was present in the input sentences. Additionally, we also intended to have an idea of whether our algorithms tended to overfit on specific entities or not. Thus, we provided a completely Unmasked version of our data and a Named Entity Recognition (NER) masked counterpart. Since the NER-masked version will have most of the entities masked, the hope is that directly comparing both scenarios can give us an idea of whether our models do overfit entities or not and, if so, to what extent that impacts performance. Let us then jump into the analysis, where table 4.12 provides an overview of the results:

	Model			
	Baseline		ESIM	
Setting	Unmasked	NER-Masked	Unmasked	NER-Masked
NL	**40.79 ± 1.49**	33.62 ± 1.10	**49.90 ± 1.90**	42.30 ± 1.38
ZSL-O	23.52 ± 4.84	22.35 ± 4.57	33.06 ± 4.68	31.84 ± 7.19
ZSL-C	17.35 ± 4.69	16.96 ± 4.74	26.42 ± 5.67	24.08 ± 4.71
FSL-1	27.39 ± 4.53	25.44 ± 4.93	37.72 ± 7.37	33.85 ± 6.85
FSL-2	35.70 ± 6.49	31.82 ± 5.31	46.59 ± 6.85	40.70 ± 6.37
FSL-5	47.51 ± 7.93	41.53 ± 7.69	57.23 ± 7.89	51.74 ± 8.99
FSL-10	**59.24 ± 4.27**	51.83 ± 3.48	**68.40 ± 4.45**	62.46 ± 4.49
GZSL-O	11.41 ± 2.67	9.69 ± 3.96	17.90 ± 3.52	17.36 ± 3.05
GZSL-C	1.15 ± 1.43	0.96 ± 1.13	2.36 ± 2.70	2.30 ± 2.56
GFSL-1	7.75 ± 5.64	4.97 ± 5.12	16.74 ± 6.05	11.80 ± 4.88
GFSL-2	14.03 ± 6.95	9.52 ± 5.85	**24.24 ± 6.18**	17.58 ± 7.34
GFSL-5	**26.96 ± 5.74**	21.03 ± 6.20	**35.42 ± 6.15**	28.88 ± 6.53
GFSL-10	**34.56 ± 4.26**	27.52 ± 6.50	**47.06 ± 4.96**	37.39 ± 5.25

Table 4.12: FE Masking - Comparing Performance Between Unmasked and NER-masked Versions

This table shows, for the FE dataset, the comparison of HMF1, or MF1 for the non-Generalised cases, test scores, in %, between the unmasked and NER-masked versions of our dataset, for both our baseline and our ESIM variant (ESIM_StS). Bold face results indicate the best result, between the unmasked and the NER-masked cases (independently evaluated for each model), with two-tailed t-test statistical significance, at a p-value of 0.05.

We start by analysing the second question, as it allows us to better discuss the first one. Concretely, both models perform better, on literally all settings, when trained and evaluated under the unmasked paradigm. Even though not all results are statistically significant, there are enough

to indicate that this seems to be a general trend. This means that the models actually benefit from having direct access to the entities, as opposed to, instead, overfitting on them. We believe that this might have to do with the models somehow understanding general properties of the entities in question, such as entity_1 being a location, for example, which can help in reducing the set of admissible relations, thus improving the chances of the model correctly determining the correct underlying relation.

In fact, and regarding now the first question, i.e. how well do the models perform when there is no specific annotation, the models trained on the unmasked version perform almost on par with the ones from our main experiment on the subject/object annotated dataset. This seems indicate that whatever extra information is being directly extracted from the entities is informative enough to both compensate for the lack of explicit annotation and even the overall noisy labelling issue that was discussed in chapter 3.

While we do not ourselves expand on this conclusion, we do mention possible future lines of work in chapter 5.

We will now turn to an unrelated line of research, that represents our initial research efforts.

4.3.5 Exploratory Unsupervised Learning

At the beginning of this thesis our focus was actually not on (Generalised) Any-Shot Learning ((G)ASL). In fact, our first intention was to leverage the NLI formulation introduced by Obamuyide and Vlachos (2018, [46]) and try to apply it to unsupervised Relation Classification (RC). The idea stemmed from the approach taken in the work of Marcheggiani and Titov (2016, [38]).

[112] *Which means they were trying to identify K classes.*

In their work, Marcheggiani and Titov (2016, [38]) attempted to learn a fixed unknown relational schema, i.e. perform a K-clusters[112] clustering approach, where the meaning of each cluster is unknown, but hopefully identifiable by a human. They claimed that most pre-existing unsupervised RE/RC methods rely on surface or syntactic patterns, which are used directly as a relation representation or that are clustered to form relations. Furthermore, they rely on simple features, as we have previously discussed in, for example, the works of Banko et al. (2007, [6]) and Etzioni et al. (2011, [18]).

In order to address this issues they proposed learning canonical representations from scratch, instead of clustering surface forms, and to leverage generative modelling as a way to train a feature rich model. Concretely, they propose the use of a Variational Auto-Encoder (VAE)-like model (Kingma and Welling, 2014, [34]), in which, in the case of Marcheggiani and Titov

(2016, [38]), the latent space is discrete and follows a uniform prior. Their opportunistic[113] use of (amortised) Variational Inference (VI) allows them to define a variational distribution that takes as input the entire sentence $\boldsymbol{x}$, circumventing the need to explicitly define features.

In order to train their model they propose to do entity reconstruction, for the entities involved in a binary relation. They consider only sentences $\boldsymbol{x}$ for which a NER deterministic algorithm is able to find a pair of co-occurring entities, $\langle e_1, e_2 \rangle$. Their idea is that of having two prediction models, one that learns to predict e_1 given $r(?, e_2)$ and another that learns to predict e_2 given $r(e_1, ?)$. Mathematically, and assuming a uniform prior over relations $\mathrm{P}(R = r) = \frac{1}{\mathcal{R}}$, they propose to optimise[114]:

$$\mathop{\mathbb{E}}_{\mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x})} \left[\sum_{i=1}^{2} \left(\log \left(\mathrm{P}_{\boldsymbol{\theta}_i} \left(e_i | e_{-i}, r \right) \right) \right) \right] + 2 \, \mathrm{H} \left(\mathrm{q}_{\boldsymbol{\lambda}} \left(r | \boldsymbol{x} \right) \right) \tag{4.24}$$

where $\mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x})$ is their variational distribution and $\mathrm{P}_{\boldsymbol{\theta}_i}(e_i|e_{-i}, r)$ represents the probabilistic model that attempts to reconstruct entity e_i given entity e_{-i}, such that $\forall_{i \in \{1,2\}} : e_i \in \{e_1, e_2\} \wedge e_{-i} \in \{e_1, e_2\} \setminus \{e_i\}$.

Inspired by their work, we attempted to have the relation descriptions, collected by Obamuyide and Vlachos (2018, [46]), ground the clusters to their actual relation.

More specifically, for our variational distribution, $\mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x})$, we used our ESIM model, as described in 4.1.2 and 4.1.3, along with the relation descriptions $\boldsymbol{y}$, to estimate a categorical distribution over our relations. This in fact means that our variational distribution is represented as $\mathrm{q}_{\boldsymbol{\lambda}}\left(r|\boldsymbol{x}, \mathcal{D}^{\mathcal{R}}\right)$, where $\mathcal{R}$ is the set of relations present in the split[115] and $\mathcal{D}^{\mathcal{R}}$ its associated set of descriptions.

Disregarding, for now, our model's decoder, we define the desired optimisation objective to be the maximization of the log-likelihood of the reconstruction of a categorical Bag of Words (BoW) representation of our input sentence $\boldsymbol{x}$. Mathematically, for a decoder $\mathrm{P}_{\theta}(\cdot)$ and a uniform prior over relations $\mathrm{P}(R = r) = \frac{1}{\mathcal{R}}$, our optimisation objective, for a single instance of a minibatch $\mathcal{M}$, is[116]:

$$\log \left(\mathrm{P} \left(\boldsymbol{x} \right) \right) \geq \mathop{\mathbb{E}}_{\mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x}, \mathcal{D}^{\mathcal{M}})} \left[\sum_{l=1}^{L_x} \left(\log \left(\mathrm{P}_{\boldsymbol{\theta}} \left(\boldsymbol{x}_l | r \right) \right) \right) \right] + \mathrm{H} \left(\mathrm{q}_{\boldsymbol{\lambda}} \left(r | \boldsymbol{x}, \mathcal{D}^{\mathcal{M}} \right) \right) \tag{4.25}$$

where L_x is the length of sentence $\boldsymbol{x}$ and $\mathcal{D}^{\mathcal{M}}$ represents the descriptions sampled for minibatch $\mathcal{M}$, just as described in 4.1.3. As we made use of the subject/object masked $\boldsymbol{x}$ sentences, we assumed that each $\boldsymbol{x}$ had a unique[117] associated latent relation.

[113] *Since the latent space is a Categorical distribution, the marginalisation over it is actually is tractable.*

[114] *The full derivation of their training objective can be found on Appendix B.1.*

[115] *Note that while this approach could also be used for (G)ASL tasks we never evaluated under such settings, only under a NL setting, since we never actually got it to work.*

[116] *The full derivation of their training objective can be found on Appendix B.2.3.*

[117] *Or most likely unique, due to noise, as discussed before in chapter 3.*

Regarding our decoder we tried several different approaches, as none seemed to work. Initially, we tried to learn a representation for each of the relations' description associated with each minibatch. These representations were either encoded by the input encoding BiLSTM of ESIM or simply encoded by a separate BiLSTM. Each relation description representation was then projected, by use of a MLP, into the train split's vocabulary size[118], and turned into a probability distribution by using the softmax function. The entries of this vocabulary by relation probabilities matrix, let us call it $\boldsymbol{V} \in \mathbb{R}^{|\mathcal{R}|} \times \mathbb{R}^{|Vocab|}$, represent our $\mathrm{P}_{\boldsymbol{\theta}}(\boldsymbol{x}_l|r)$ of equation 4.25. Training was not straight forward and we had to recourse to methods such as Kullback–Leibler divergence (KL) annealing (R. Bowman et al., 2016, [53]) and inducing a sparse prior over relations, by sampling the prior parameters from a Dirichlet distribution.

[118] *Note that we are not concerned with any of the other splits' vocabulary, as we only need the decoder during training, since the whole point is to train a RC model, that is in fact our encoder.*

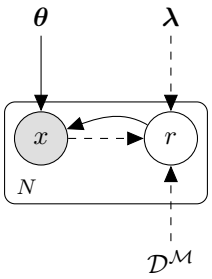


Figure 4.18: Unsupervised Approach's Graphical Model

However, after struggling with this formulation for a while we eventually realised that there was no benefit whatsoever in predicting $\boldsymbol{V}$ from the relation descriptions.

In fact, since each description $\boldsymbol{y}$ is paired with a specific relation r, $\boldsymbol{y}$ itself contains no more information about $\boldsymbol{x}$ than r does. This indicates that we could simply associate each row of $\boldsymbol{V}$ with a specific relation r, regardless of whatever description was used to represent r.

Accordingly, we decided to model $\boldsymbol{V}$ directly, as a matrix to be learned. We were also not successful, as our posterior distribution quickly collapsed in such a way that it learned to predict a single relation.

We believe that this is due to the sheer number of words present in the vocabulary, for which we try to predict a categorical distribution given each individual relation. As the majority of the words in the vocabulary will, most likely, not be informative about the underlying relation in $\boldsymbol{x}$, we think the model optimised the prediction of these words at the start of training. It then reduces its own complexity by making use of a single relation (as the non-relation-informative words are independent of a specific relation, any of the rows in $\boldsymbol{V}$ can be used to model these). However, by the time it has learned the probability associated with these non-relation-informative words the posterior has collapsed in such a way that makes it hard to have a gradient flow that would help in modelling the other distinct relations.

We tried to compensate for these by, again, inducing a sparse prior over the relations, that was different for each minibatch, by sampling the prior parameters from Dirichlet distribution. A more natural way to address the problem would have been to impose the prior directly on $\boldsymbol{V}$, but this is left for future work. We also tried to pretrain the decoder, $\boldsymbol{V}$, so that each row, which is associated with a specific relation, would be biased towards predicting the words present in the corresponding relation's descriptions. This

would hopefully guide the decoder in better matching each latent relation with the desired corresponding set of descriptions. Finally, we attempted to leverage dropout (Srivastava et al., 2014, [65]) as formulated by Gal and Ghahramani (2015, [21]). Yet, none of these approaches yielded the desired results.

All the experiments described in this subsection (that is, in 4.3.5) were performed quite early on into this thesis. However, as none of our approaches seemed to be even remotely successful, and it seemed like this could be a problem that would take quite some time to solve, we quickly decided to instead divert our attention to the supervised case and the study of (Generalised) Any-Shot Learning ((G)ASL). This ended up becoming the main topic of this thesis and representing the bulk of the undertaken work. Even so, these unsupervised experiments were the initial driving motivation of framing our ESIM variant as a direct categorical estimation over the relevant relations, as that enabled us to leverage the framework of (amortised) Variational Inference (VI). This ended up, coincidently, aligning nicely with what became our final approach.

Chapter 5

Conclusion and Future Work

In this chapter we will have a final look at the insights that we were able to draw during this thesis and also discuss possible future lines of work that would be directly related to what we did.

5.1 Conclusion

Now that this thesis has come to an end, what can we conclude from it?

Perhaps the most important contribution was the study of the Generalised aspect of (Generalised) Any-Shot Learning ((G)ASL). With this, our intention was to clearly demonstrate that the plain ASL aspect of the task is overly simplistic, rendering the analysis of methods' performance on previously *unseen* classes unrealistic. To this end, we have discussed the use of the more adequate Harmonic Macro F1-Score metric for GASL and also successfully created new splits for the task of Relation Classification (RC), which will hopefully provide a common evaluation ground for future research, rendering it more comparable and realistic.

Still on GASL, we have shown that when performing (Generalised) Zero-Shot Learning ((G)ZSL) tasks it is beneficial to frame the training procedure under an Open Set framework, as opposed to a Closed Set framework.

Additionally, we have also evaluated how well a Natural Language Inference (NLI) scoring function based model can perform on the aforementioned (G)ASL tasks. In specific, our NLI formulation represents a more direct approach, contrasting with independent binary classification methods of previous works. We have also shown that more complex proven architectures bring an advantage to these tasks, over simpler models, such as our BiLSTM baseline.

Finally, we have also shown that our NLI based model can also perform quite well when faced with no specific entity annotation whatsoever, potentially paving the work for future methods, which we shall quickly discuss in the next section.

5.2 Future Work

In this section we will quickly discuss some potential future lines of work.

Borrowing from Computer Vision Literature

As we have mentioned earlier, the CV branch of AI has devoted more resources then NLP to the study of (G)ASL. As such, there are a significant number of works that could potentially be usefully adapted for future NLP research. For the sake of brevity we choose to highlight only one that we found particularly interesting.

Based on **ReViSE** (Tsai et al., 2017, [66]), a method that employs a contractive AE (Rifai et al., 2011, [57]) for the visual input data and a vanilla AE (Hinton and Salakhutdinov, 2006, [29]) for the class-level text data[119], **CADA-VAE** (Schönfeld et al., 2019, [60]) replaces the AEs with VAEs (Kingma and Welling, 2014, [34]). This, in turn, allows them to leverage the classes side information to generate latent data[120] for the *unseen* classes. They train a classifier directly on the latent space[121], making use of the *unseen* classes' generated latent codes to help in addressing the data imbalance problem, which enabled them to have close to equal performance on both the *seen* and *unseen* classes.

Similar data generating approaches could also be extremely useful for NLP task and methods, and we think it would be very interesting to explore such an approach.

Multi Task Learning

As our model is task agnostic[122], we think it would be particularly interesting to see if performance would improve when training for multiple distinct tasks. This could potentially be achieved by interleaving minibatches from each of the desired tasks, which would hopefully induce a more general, and beneficial, NLI capability in the models. Since we have already demonstrated that our model can perform quite well on completely unannotated data, this makes this Multi Task Learning investigation potentially easier.

More Advanced NLI Architectures

A simple and obvious future line of research would be to evaluate how different, more advanced NLI based architectures would perform on Generalised Any-Shot Learning (GASL) tasks.

[119] *And attempt to associate one modality with the other, i.e. match both embedding spaces, through the Maximum Mean Discrepancy (MMD) (Gretton et al., 2007, [25]) criterion. The use of AEs also allows them to use unlabelled data.*

[120] *That is, in the form of latent codes.*

[121] *This means they restrict themselves to a Closed Set Framework.*

[122] *That is, it can be applied to other textual classification tasks besides RC.*

Assessing the Impact of Information Leaking due to the use of Pretrained Embedding Models

Like we mentioned in 3.2.2, pretrained embedding models can potentially have access to data regarding the *unseen* classes, which would be preferably avoidable. This could potentially not be of significant impact, as these models are generally not trained on data specifically structured for the data considered in general NLP tasks. Nonetheless, we think it is important to have an idea of how impactful this actually is.

Influence of Number of Available Class Descriptions

Potentially one of the most important aspects of our approach is the ability to express classes in a way that matches Natural Language expressivity. As such, we think it is important to assess the impact of having a varying number of descriptions per class. It would also be interesting to better understand how increasing the overall number of descriptions, in a meaningful way, improves, or not, the models' distinguishing capabilities.

Better Class Descriptions

One aspect that we have not previously mentioned is that some of the descriptions created by Obamuyide and Vlachos (2018, [46]) have either some small orthographic or syntactic errors. We do not believe this is of significant importance (particularly so when using contextualized embeddings, as we do), but it would still be interesting to assess how the quality of the descriptions affects overall model performance.

Expand our Model to Consider the Actual Entities

As we have shown in 4.3.4 that our model seems to benefit a lot from having direct assess to entities involved in a relation, we think that modifying it to directly take into account these entities could potentially improve its performance.

Relation Extraction Informative Model

Since our model computes a score for all known relations, we think it could potentially be used as first step in RE, by estimating which relation are most likely present in a specific sentence. These could both benefit the identification of relevant entities and also significantly reduce the admissible set of possible relations between any one entity pair.

Unsupervised Relation Classification

Regarding our unsupervised learning experiments (4.3.5), we feel that the approach of directly imposing a prior on $\boldsymbol{V}$, could be a next step to try to address the issues discussed in that section.

Bibliography

[1] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-embedding for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7):1425–1438, July 2016. ISSN 0162-8828. doi: 10.1109/TPAMI.2015.2487986.

[2] Elena Akhmatova. Textual entailment resolution via atomic propositions. Citeseer, 2005.

[3] Ziad Al-Halah, Makarand Tapaswi, and Rainer Stiefelhagen. Recovering the missing link: Predicting class-attribute associations for unsupervised zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5975–5984, 2016.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

[5] Jorge Balazs, Edison Marrese-Taylor, Pablo Loyola, and Yutaka Matsuo. Refining raw sentence representations for textual entailment recognition via attention. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 51–55, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5310. URL https://www.aclweb.org/anthology/W17-5310.

[6] Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *Ijcai*, volume 7, pages 2670–2676, 2007.

[7] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM, 2008.

[8] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[9] Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. Synthesized classifiers for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5327–5336, 2016.

[10] Wei-Lun Chao, Soravit Changpinyo, Boqing Gong, and Fei Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *European Conference on Computer Vision*, pages 52–68. Springer, 2016.

[11] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada, July

2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1152. URL https://www.aclweb.org/anthology/P17-1152.

[12] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 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*, pages 36–40, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5307. URL https://www.aclweb.org/anthology/W17-5307.

[13] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1070. URL https://www.aclweb.org/anthology/D17-1070.

[14] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, January 1967. ISSN 0018-9448. doi: 10.1109/TIT.1967.1053964.

[15] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In Joaquin Quiñonero-Candela, Ido Dagan, Bernardo Magnini, and Florence d'Alché Buc, editors, *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-33428-6.

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[18] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, et al. Open information extraction: The second generation. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[19] Abraham Fowler, Bob Hauser, Daniel Hodges, Ian Niles, Adrian Novischi, and Jens Stephan. Applying cogex to recognize textual entailment. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 69–72. Citeseer, 2005.

[20] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129, 2013.

[21] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2015.

[22] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. 2017.

[23] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[24] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602 – 610, 2005. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2005.06.042. URL http://www.sciencedirect.com/science/article/pii/S0893608005001206. IJCNN 2005.

[25] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J. Smola. A kernel method for the two-sample-problem. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 513–520. MIT Press, 2007. URL http://papers.nips.cc/paper/3110-a-kernel-method-for-the-two-sample-problem.pdf.

[26] Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. Fewrel:a large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *EMNLP*, 2018.

[27] Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, SEW '09, pages 94–99, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-31-2. URL http://dl.acm.org/citation.cfm?id=1621969.1621986.

[28] Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. WikiReading: A novel large-scale language understanding task over Wikipedia. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1545, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1145. URL https://www.aclweb.org/anthology/P16-1145.

[29] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9: 1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[31] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

[32] Johan Ludwig William Valdemar Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 30(1):175–193, 1906.

[33] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[34] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

[35] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2013.

[36] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. Zero-data learning of new tasks. In *AAAI*, volume 1, page 3, 2008.

[37] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342. Association for Computational Linguistics, 2017. doi: 10.18653/v1/K17-1034. URL http://aclweb.org/anthology/K17-1034.

[38] Diego Marcheggiani and Ivan Titov. Discrete-state variational autoencoders for joint discovery and factorization of relations. *Transactions of the Association for Computational Linguistics*, 4: 231–244, 2016.

[39] Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*, 2013, 01 2013.

[40] Erik G Miller, Nicholas E Matsakis, and Paul A Viola. Learning from one example through shared densities on transforms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 1, pages 464–471. IEEE, 2000.

[41] Erik Gundersen Miller. *Learning from one example in machine vision by sharing probability densities*. PhD thesis, Massachusetts Institute of Technology, 2002.

[42] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009.

[43] Tsendsuren Munkhdalai and Hong Yu. Neural semantic encoders. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 397–407, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E17-1038.

[44] Yixin Nie and Mohit Bansal. Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 41–45, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5308. URL https://www.aclweb.org/anthology/W17-5308.

[45] Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*, 2013.

[46] Abiola Obamuyide and Andreas Vlachos. Zero-shot relation classification as textual entailment. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 72–78, 2018.

[47] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1410–1418. Curran Associates, Inc., 2009. URL http://papers.nips.cc/paper/3650-zero-shot-learning-with-semantic-output-codes.pdf.

[48] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1244. URL https://www.aclweb.org/anthology/D16-1244.

[49] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[50] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

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

[52] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

[53] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. pages 10–21, 01 2016. doi: 10.18653/v1/K16-1002.

[54] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[55] Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer, 2010.

[56] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, 2013.

[57] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 833–840. Omnipress, 2011.

[58] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomas Kocisky, and Phil Blunsom. Reasoning about entailment with neural attention. In *International Conference on Learning Representations (ICLR)*, 2016.

[59] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(7): 1757–1772, 2012.

[60] Edgar Schönfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. Generalized zero-and few-shot learning via aligned variational autoencoders. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8247–8255, 2019.

[61] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[62] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264, 2006.

[63] Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. Matching the blanks: Distributional similarity for relation learning. *arXiv preprint arXiv:1906.03158*, 2019.

[64] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943, 2013.

[65] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[66] Yao-Hung Hubert Tsai, Liang-Kang Huang, and Ruslan Salakhutdinov. Learning robust visual-semantic embeddings. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3591–3600. IEEE, 2017.

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

[68] Denny Vrandečić. Wikidata: A new platform for collaborative data collection. In *Proceedings of the 21st international conference on world wide web*, pages 1063–1064. ACM, 2012.

[69] Shuohang Wang and Jing Jiang. 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*, pages 1442–1451, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1170. URL https://www.aclweb.org/anthology/N16-1170.

[70] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4144–4150, 2017. doi: 10.24963/ijcai.2017/579. URL https://doi.org/10.24963/ijcai.2017/579.

[71] Adina Williams, Nikita Nangia, and Samuel Bowman. 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)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL https://www.aclweb.org/anthology/N18-1101.

[72] Yongqin Xian, Zeynep Akata, Gaurav Sharma, Quynh Nguyen, Matthias Hein, and Bernt Schiele. Latent embeddings for zero-shot classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 69–77, 2016.

[73] Yongqin Xian, Bernt Schiele, and Zeynep Akata. Zero-shot learning-the good, the bad and the ugly. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4582–4591, 2017.

[74] Majid Yazdani and James Henderson. A model of zero-shot learning of spoken language understanding. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 244–249, 2015.

[75] Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 35–45, 2017. URL https://nlp.stanford.edu/pubs/zhang2017tacred.pdf.

[76] Ziming Zhang and Venkatesh Saligrama. Zero-shot learning via semantic similarity embedding. In *Proceedings of the IEEE international conference on computer vision*, pages 4166–4174, 2015.

Appendices

A Additional Tables

A.1 Overlapping Relation Descriptions

Relation Description	(#Relations) - Relations
OBJECT_ENTITY created SUBJECT_ENTITY	(5) - founder, architect, performer, editor, author
OBJECT_ENTITY is the name of the director of SUBJECT_ENTITY	(2) - film editor, director
OBJECT_ENTITY had the role of director in SUBJECT_ENTITY	(2) - film editor, director
OBJECT_ENTITY directed the film SUBJECT_ENTITY	(2) - film editor, director
OBJECT_ENTITY is the director of SUBJECT_ENTITY	(3) - film editor, screenwriter, director
OBJECT_ENTITY was responsible for the direction of SUBJECT_ENTITY	(3) - film editor, screenwriter, director
OBJECT_ENTITY directed SUBJECT_ENTITY	(3) - film editor, screenwriter, director
The director of SUBJECT_ENTITY is OBJECT_ENTITY	(2) - film editor, director
OBJECT_ENTITY was the director of SUBJECT_ENTITY	(3) - film editor, screenwriter, director
OBJECT_ENTITY was when SUBJECT_ENTITY was created	(2) - date of official opening, inception
OBJECT_ENTITY is the position of SUBJECT_ENTITY	(2) - position played on team / speciality, position held
OBJECT_ENTITY developed SUBJECT_ENTITY	(2) - designer, creator
OBJECT_ENTITY designed SUBJECT_ENTITY	(2) - designer, architect
SUBJECT_ENTITY was designed by OBJECT_ENTITY	(2) - designer, architect
OBJECT_ENTITY worked on SUBJECT_ENTITY	(2) - designer, developer
OBJECT_ENTITY was the designer of SUBJECT_ENTITY	(2) - designer, architect
OBJECT_ENTITY made SUBJECT_ENTITY	(2) - manufacturer, performer
OBJECT_ENTITY is the creator of SUBJECT_ENTITY	(2) - creator, author
The song SUBJECT_ENTITY was by OBJECT_ENTITY	(2) - performer, lyrics by
OBJECT_ENTITY recorded SUBJECT_ENTITY	(2) - performer, lyrics by
OBJECT_ENTITY was the nationality of SUBJECT_ENTITY	(2) - native language:, languages spoken or written
The nationality of SUBJECT_ENTITY is OBJECT_ENTITY	(2) - native language:, languages spoken or written
The film SUBJECT_ENTITY was directed by OBJECT_ENTITY	(2) - screenwriter, director
OBJECT_ENTITY was the director for SUBJECT_ENTITY	(2) - screenwriter, director
OBJECT_ENTITY served as director for SUBJECT_ENTITY	(2) - screenwriter, director
SUBJECT_ENTITY was directed by OBJECT_ENTITY	(2) - screenwriter, director
OBJECT_ENTITY wrote SUBJECT_ENTITY	(2) - lyrics by, author
OBJECT_ENTITY was the year SUBJECT_ENTITY was created	(2) - inception, publication date
OBJECT_ENTITY was the cause of death of SUBJECT_ENTITY	(2) - cause of death, manner of death
OBJECT_ENTITY caused SUBJECT_ENTITY's death	(2) - cause of death, medical condition

Table A.1: Overlapping Relation Descriptions

List of relation descriptions, as created by Obamuyide and Vlachos (2018, [46]), that are present in multiple relations. That is, relation descriptions that actually describe multiple relations.

A.2 Number of Relation Descriptions

Relation	# Descriptions	Relation	# Descriptions
airline hub	9	located next to body of water	3
architect	29	located on astronomical body	9
architectural style	3	location of formation	6
author	27	lyrics by	12
award received	5	manner of death	4
based on	2	manufacturer	15
brother	2	material used	2
canonization status	1	medical condition	18
cast member	10	member of political party	7
cause of death	23	member of sports team	15
chairperson	1	military branch	26
characters	1	military rank	2
child	2	mother	9
chromosome	10	mouth of the watercourse	9
collection	1	named after	4
conferred by	1	narrative location	2
conflict	20	native language	10
connecting line	11	noble family	3
constellation	21	noble title	2
continent	18	nominated for	2
convicted of	3	occupant	1
country	22	occupation	3
country of citizenship	16	operating system	1
country of origin	6	original network	34
creator	7	parent company	3
crosses	4	parent taxon	5
date of birth	9	participant of	9
date of death	20	performer	29
date of official opening	14	place of birth	14
designer	9	place of burial	11
developer	9	place of death	5
director	23	point in time	12
discoverer or inventor	15	position held	3
dissolved or abolished	9	position played on team / speciality	28
distributor	19	present in work	10
drafted by	6	product	6
editor	4	production company	5
educated at	8	programming language	3
employer	1	publication date	28
end time	1	publisher	7
father	14	record label	4
film editor	17	religious order	2
found in taxon	3	replaced by	3
founder	9	residence	2
from fictional universe	11	screenwriter	14
head of government	4	series	14
headquarters location	6	service entry	10
home venue	29	sex or gender	20
illustrator	4	sister	2
inception	26	site of astronomical discovery	9
industry	1	sport	17
instrument	24	spouse	1
instrumentation	15	standards body	1
IUCN conservation status	8	start time	1
language of work or name	8	stock exchange	9
languages spoken or written	8	taxon rank	2
league	8	time of discovery	19
license	1	time of spacecraft launch	12
licensed to broadcast to	3	vessel class	4
located in the administrative territorial entity	3	voice type	8

Table A.2: Number of Descriptions per Relation

A.3 1 vs 2 BiLSTM(s) - The Manifold Matching Problem

Hidden Layer Size	NL
50	-1.33 ± 4.24
100	-3.50 ± 3.94
200	-0.21 ± 2.17
300	-0.46 ± 3.39
400	-0.90 ± 4.15
500	-0.78 ± 2.59

Table A.3: Difference of using 1 vs 2 BiLSTM(s) (Baseline - NL setting)

This table shows, for the NL setting, the differences of using a single BiLSTM vs using two BiLSTMs, i.e. we compute metric_diff(1 BiLSTM, 2 BiLSTMs) for each HT fold and then average over folds. All other hyperparameters are fixed. Bold face results indicate statistical significance at a p-value of 0.05, under a t-test (in this case no differences were statistical significant).

Hidden Layer Size	ZS		GZS	
	O-Set	C-Set	O-Set	C-Set
50	6.55 ± 6.69	**8.47 ± 3.04**	5.10 ± 6.13	1.45 ± 1.99
100	6.30 ± 6.54	**10.47 ± 4.38**	3.54 ± 5.79	3.05 ± 4.20
200	5.62 ± 7.05	5.00 ± 8.04	6.24 ± 14.36	1.18 ± 1.73
300	7.87 ± 10.19	**13.22 ± 7.88**	**15.55 ± 9.97**	0.61 ± 0.94
400	10.36 ± 5.82	3.09 ± 10.10	**14.86 ± 4.81**	1.54 ± 4.32
500	8.70 ± 13.24	8.27 ± 10.44	5.08 ± 2.89	-1.69 ± 3.71

Table A.4: Difference of using 1 vs 2 BiLSTM(s) (Baseline - (G)ZSL settings)

This table shows, for our baseline and the (G)ZSL settings, the differences of using a single BiLSTM vs using two BiLSTMs, i.e. we compute metric_diff(1 BiLSTM, 2 BiLSTMs) for each HT fold and then average over folds. All other hyperparameters are fixed. Bold face results indicate statistical significance at a p-value of 0.05, under a t-test.

Hidden Layer Size	FS			
	1	2	5	10
50	6.03 ± 5.49	7.31 ± 5.56	-2.19 ± 5.35	1.20 ± 4.64
100	4.41 ± 6.62	4.16 ± 11.19	-0.24 ± 7.67	4.16 ± 6.45
200	4.02 ± 8.44	1.61 ± 3.89	0.50 ± 5.20	0.81 ± 4.04
300	5.17 ± 5.11	5.22 ± 4.73	1.69 ± 9.88	-7.04 ± 6.13
400	4.40 ± 9.21	0.51 ± 4.74	7.23 ± 9.31	-2.32 ± 5.53
500	10.02 ± 7.58	2.88 ± 6.13	-0.27 ± 6.01	4.11 ± 4.50

Table A.5: Difference of using 1 vs 2 BiLSTM(s) (Baseline - FSL settings)

This table shows, for our baseline and the FSL settings, the differences of using a single BiLSTM vs using two BiLSTMs, i.e. we compute metric_diff(1 BiLSTM, 2 BiLSTMs) for each HT fold and then average over folds. All other hyperparameters are fixed. Bold face results indicate statistical significance at a p-value of 0.05, under a t-test (in this case no differences were statistical significant).

Hidden Layer Size	GFS			
	1	2	5	10
50	3.58 ± 5.26	**8.78 ± 7.08**	-0.24 ± 6.18	3.35 ± 7.44
100	5.77 ± 5.60	10.76 ± 6.26	-3.07 ± 5.80	3.44 ± 5.16
200	**9.69 ± 8.04**	6.97 ± 9.06	2.89 ± 7.91	-0.70 ± 6.26
300	9.08 ± 11.26	12.01 ± 5.12	4.00 ± 9.82	0.50 ± 6.54
400	**9.71 ± 5.96**	7.43 ± 9.36	0.72 ± 5.99	5.86 ± 5.73
500	**10.25 ± 10.37**	4.28 ± 6.13	4.85 ± 5.15	1.75 ± 3.43

Table A.6: Difference of using 1 vs 2 BiLSTM(s) (Baseline - GFSL settings)

This table shows, for our baseline and the GFSL settings, the differences of using a single BiLSTM vs using two BiLSTMs, i.e. we compute metric_diff(1 BiLSTM, 2 BiLSTMs) for each HT fold and then average over folds. All other hyperparameters are fixed. Bold face results indicate statistical significance at a p-value of 0.05, under a t-test.

B Mathematical Derivations

B.1 General Entity Argument Reconstruction Training Objective

Following the notation of Marcheggiani and Titov (2016, [38]), which we revisit here: for $i \in \{1, 2\}$ e_i is one of the two entities found by a NER algorithm and e_{-i} is the complement entity, i.e. $\forall_{i \in \{1,2\}}$: $e_i \in \{e_1, e_2\} \wedge e_{-i} \in \{e_1, e_2\} \setminus \{e_i\}$.

For a set of parameters $\mathbf{\Theta} = \mathbf{\Theta}_1 \cup \mathbf{\Theta}_2$, where $\mathbf{\Theta}_i = \{\boldsymbol{\theta}_i, \boldsymbol{\lambda}_i\}$, the point is to find $\mathbf{\Theta}$ that maximises $\mathrm{P}_{\mathbf{\Theta}_1}(e_1|e_2)$ and $\mathrm{P}_{\mathbf{\Theta}_2}(e_2|e_1)$ simultaneously, which can be seen as:

$$\arg\max_{\mathbf{\Theta}} \left(\prod_{i=1}^{2} \mathrm{P}_{\mathbf{\Theta}_i}(e_i|e_{-i}) \right) = \arg\max_{\mathbf{\Theta}} \left(\log \left(\prod_{i=1}^{2} \mathrm{P}_{\mathbf{\Theta}_i}(e_i|e_{-i}) \right) \right) =$$

$$= \arg\max_{\mathbf{\Theta}} \left(\sum_{i=1}^{2} \log \left(\mathrm{P}_{\mathbf{\Theta}_i}(e_i|e_{-i}) \right) \right) \tag{B.1.1}$$

We drop the $\arg\max_{\mathbf{\Theta}}$ to avoid clutter and introduce one variational distribution per model, $\mathrm{q}_{R,\boldsymbol{\lambda}_i}(r)$, which we simply write as $\mathrm{q}_{\boldsymbol{\lambda}_i}(r)$. We continue the derivation following two different methods:

1. Jensen's Inequality

$$(B.1.1) = \sum_{i=1}^{2} \log \left(\sum_{r \in \mathcal{R}} \mathrm{P}_{\mathbf{\Theta}_i}(e_i, r|e_{-i}) \right) = \sum_{i=1}^{2} \log \left(\sum_{r \in \mathcal{R}} \frac{\mathrm{q}_{\boldsymbol{\lambda}_i}(r)}{\mathrm{q}_{\boldsymbol{\lambda}_i}(r)} \mathrm{P}_{\mathbf{\Theta}_i}(e_i, r|e_{-i}) \right) =$$

$$= \sum_{i=1}^{2} \log \left(\sum_{r \in \mathcal{R}} \mathrm{q}_{\boldsymbol{\lambda}_i}(r) \frac{\mathrm{P}_{\mathbf{\Theta}_i}(e_i, r|e_{-i})}{\mathrm{q}_{\boldsymbol{\lambda}_i}(r)} \right) \tag{B.1.2}$$

Since the logarithm is a concave monotonic function, we make use of Jensen's Inequality (Jensen, 1906, [32]):

$$(B.1.2) \geq \sum_{i=1}^{2} \sum_{r \in \mathcal{R}} \mathrm{q}_{\boldsymbol{\lambda}_i}(r) \log \left(\frac{\mathrm{P}_{\mathbf{\Theta}_i}(e_i, r|e_{-i})}{\mathrm{q}_{\boldsymbol{\lambda}_i}(r)} \right) = \sum_{i=1}^{2} \sum_{r \in \mathcal{R}} \mathrm{q}_{\boldsymbol{\lambda}_i}(r) \log \left(\frac{\mathrm{P}_{\boldsymbol{\theta}_i}(e_i|e_{-i}, r)\, \mathrm{P}(r|e_{-i})}{\mathrm{q}_{\boldsymbol{\lambda}_i}(r)} \right) =$$

$$= \sum_{i=1}^{2} \left(\sum_{r \in \mathcal{R}} \mathrm{q}_{\boldsymbol{\lambda}_i}(r) \log \left(\mathrm{P}_{\boldsymbol{\theta}_i}(e_i|e_{-i}, r) \right) + \sum_{r \in \mathcal{R}} \mathrm{q}_{\boldsymbol{\lambda}_i}(r) \log \left(\frac{\mathrm{P}(r|e_{-i})}{\mathrm{q}_{\boldsymbol{\lambda}_i}(r)} \right) \right) =$$

$$= \sum_{i=1}^{2} \left(\mathop{\mathbb{E}}_{\mathrm{q}_{\boldsymbol{\lambda}_i}(r)} \left[\log \left(\mathrm{P}_{\boldsymbol{\theta}_i}(e_i|e_{-i}, r) \right) \right] - \mathrm{KL}\left(\mathrm{q}_{\boldsymbol{\lambda}_i}(r) \,||\, \mathrm{P}(r|e_{-i}) \right) \right) \tag{B.1.3}$$

2. Dropping KL between Variational Distribution and Posterior

$$
\begin{aligned}
(B.1.1) &= \sum_{i=1}^{2} \log\left(\mathrm{P}_{\boldsymbol{\Theta}_i}\left(e_i|e_{-i}\right)\right) \overbrace{\sum_{r\in\mathcal{R}} \mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right)}^{1} = \sum_{i=1}^{2}\sum_{r\in\mathcal{R}} \mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right) \log\left(\mathrm{P}_{\boldsymbol{\Theta}_i}\left(e_i|e_{-i}\right)\right) = \\
&= \sum_{i=1}^{2}\sum_{r\in\mathcal{R}} \mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right) \log\left(\mathrm{P}_{\boldsymbol{\Theta}_i}\left(e_i|e_{-i}\right) \frac{\mathrm{P}\left(e_i, r|e_{-i}\right)}{\mathrm{P}\left(e_i, r|e_{-i}\right)} \frac{\mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right)}{\mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right)}\right) = \\
&= \sum_{i=1}^{2}\sum_{r\in\mathcal{R}} \mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right) \log\left(\cancel{\mathrm{P}_{\boldsymbol{\Theta}_i}\left(e_i|e_{-i}\right)} \frac{\mathrm{P}_{\boldsymbol{\theta}_i}\left(e_i|e_{-i}, r\right)\mathrm{P}\left(r|e_{-i}\right)}{\mathrm{P}\left(r|e_i, e_{-i}\right)\cancel{\mathrm{P}_{\boldsymbol{\Theta}_i}\left(e_i|e_{-i}\right)}} \frac{\mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right)}{\mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right)}\right) = \\
&= \sum_{i=1}^{2}\sum_{r\in\mathcal{R}} \mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right) \log\left(\mathrm{P}_{\boldsymbol{\theta}_i}\left(e_i|e_{-i}, r\right) \frac{\mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right)}{\mathrm{P}\left(r|e_i, e_{-i}\right)} \frac{\mathrm{P}\left(r|e_{-i}\right)}{\mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right)}\right) = \\
&= \sum_{i=1}^{2}\sum_{r\in\mathcal{R}} \mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right) \left(\log\left(\mathrm{P}_{\boldsymbol{\theta}_i}\left(e_i|e_{-i}, r\right)\right) + \log\left(\frac{\mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right)}{\mathrm{P}\left(r|e_i, e_{-i}\right)}\right) + \log\left(\frac{\mathrm{P}\left(r|e_{-i}\right)}{\mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right)}\right)\right) = \\
&= \sum_{i=1}^{2}\sum_{r\in\mathcal{R}} \mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right) \left(\log\left(\mathrm{P}_{\boldsymbol{\theta}_i}\left(e_i|e_{-i}, r\right)\right) + \log\left(\frac{\mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right)}{\mathrm{P}\left(r|e_i, e_{-i}\right)}\right) - \log\left(\frac{\mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right)}{\mathrm{P}\left(r|e_{-i}\right)}\right)\right) = \\
&= \sum_{i=1}^{2} \left(\mathop{\mathbb{E}}_{\mathrm{q}_{\boldsymbol{\lambda}_i}(r)}\left[\log\left(\mathrm{P}_{\boldsymbol{\theta}_i}\left(e_i|e_{-i}, r\right)\right)\right] + \overbrace{\mathrm{KL}\left(\mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right)||\mathrm{P}\left(r|e_i, e_{-i}\right)\right)}^{\geq 0} - \mathrm{KL}\left(\mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right)||\mathrm{P}\left(r|e_{-i}\right)\right)\right) \geq \\
&\geq \sum_{i=1}^{2} \left(\mathop{\mathbb{E}}_{\mathrm{q}_{\boldsymbol{\lambda}_i}(r)}\left[\log\left(\mathrm{P}_{\boldsymbol{\theta}_i}\left(e_i|e_{-i}, r\right)\right)\right] - \mathrm{KL}\left(\mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right)||\mathrm{P}\left(r|e_{-i}\right)\right)\right) \qquad \text{(B.1.4)}
\end{aligned}
$$

In the specific case of Marcheggiani and Titov (2016, [38]) they reduce both variational distributions, $\mathrm{q}_{\boldsymbol{\lambda}_i}\left(r\right)$, to a single one $\mathrm{q}_{\boldsymbol{\lambda}}\left(r\right)$.

B.2 Categorical Bag of Words Sentence Reconstruction Training Objective

For a minibatch $\mathcal{M}$ of sentences $\boldsymbol{x}$ and a set of parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}, \boldsymbol{\lambda}\}$:

$$
\begin{aligned}
&\arg\max_{\boldsymbol{\Theta}} \left(\prod_{m=1}^{|\mathcal{M}|} \mathrm{P}_{\boldsymbol{\Theta}}\left(\boldsymbol{x}_m\right)\right) = \arg\max_{\boldsymbol{\Theta}} \left(\log\left(\prod_{m=1}^{|\mathcal{M}|} \mathrm{P}_{\boldsymbol{\Theta}}\left(\boldsymbol{x}_m\right)\right)\right) = \\
&= \arg\max_{\boldsymbol{\Theta}} \left(\sum_{m=1}^{|\mathcal{M}|} \log\left(\mathrm{P}_{\boldsymbol{\Theta}}\left(\boldsymbol{x}_m\right)\right)\right) \qquad \text{(B.2.1)}
\end{aligned}
$$

We drop the $\arg\max_{\boldsymbol{\Theta}}$ to avoid clutter and introduce a variational distributionl, $\mathrm{q}_{R,\boldsymbol{\lambda}}\left(r|\boldsymbol{x}_m\right)$, which

we simply write as $\mathrm{q}_{\boldsymbol{\lambda}}(r)$.

$$(B.2.1) = \sum_{m=1}^{|\mathcal{M}|} \log\left(\sum_{r\in\mathcal{R}} \mathrm{P}_{\boldsymbol{\Theta}}(\boldsymbol{x}_m, r)\right) = \sum_{m=1}^{|\mathcal{M}|} \log\left(\sum_{r\in\mathcal{R}} \frac{\mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x}_m)}{\mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x}_m)} \mathrm{P}_{\boldsymbol{\Theta}}(\boldsymbol{x}_m, r)\right) =$$

$$= \sum_{m=1}^{|\mathcal{M}|} \log\left(\sum_{r\in\mathcal{R}} \mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x}_m) \frac{\mathrm{P}_{\boldsymbol{\Theta}}(\boldsymbol{x}_m, r)}{\mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x}_m)}\right) \tag{B.2.2}$$

Since the logarithm is a concave monotonic function, we make use of Jensen's Inequality (Jensen, 1906, [32]):

$$(B.2.2) \geq \sum_{m=1}^{|\mathcal{M}|} \sum_{r\in\mathcal{R}} \mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x}_m) \log\left(\frac{\mathrm{P}_{\boldsymbol{\Theta}}(\boldsymbol{x}_m, r)}{\mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x}_m)}\right) = \sum_{m=1}^{|\mathcal{M}|} \sum_{r\in\mathcal{R}} \mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x}_m) \log\left(\frac{\mathrm{P}_{\boldsymbol{\theta}}(\boldsymbol{x}_m|r)\,\mathrm{P}(r)}{\mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x}_m)}\right) =$$

$$= \sum_{m=1}^{|\mathcal{M}|} \left(\sum_{r\in\mathcal{R}} \mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x}_m) \log\left(\mathrm{P}_{\boldsymbol{\theta}}(\boldsymbol{x}_m|r)\right) + \sum_{r\in\mathcal{R}} \mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x}_m) \log\left(\frac{\mathrm{P}(r)}{\mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x}_m)}\right)\right) =$$

$$= \sum_{m=1}^{|\mathcal{M}|} \left(\mathop{\mathbb{E}}_{\mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x}_m)} \left[\log\left(\mathrm{P}_{\boldsymbol{\theta}}(\boldsymbol{x}_m|r)\right)\right] - \mathrm{KL}\left(\mathrm{q}_{\boldsymbol{\lambda}}(r)\,||\mathrm{P}(r|\boldsymbol{x})\right)\right) =$$

$$= \sum_{m=1}^{|\mathcal{M}|} \left(\mathop{\mathbb{E}}_{\mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x}_m)} \left[\log\left(\prod_{l=1}^{l_m} \mathrm{P}_{\boldsymbol{\theta}}(\boldsymbol{x}_{ml}|r)\right)\right] - \mathrm{KL}\left(\mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x})\,||\mathrm{P}(r)\right)\right) =$$

$$= \sum_{m=1}^{|\mathcal{M}|} \left(\mathop{\mathbb{E}}_{\mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x}_m)} \left[\sum_{l=1}^{l_m} \log\left(\mathrm{P}_{\boldsymbol{\theta}}(\boldsymbol{x}_{ml}|r)\right)\right] - \mathrm{KL}\left(\mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x})\,||\mathrm{P}(r)\right)\right) =$$

$$= \sum_{m=1}^{|\mathcal{M}|} \left(\sum_{l=1}^{l_m} \mathop{\mathbb{E}}_{\mathrm{q}_{\boldsymbol{\lambda}}(r|\boldsymbol{x}_m)} \left[\log\left(\mathrm{P}_{\boldsymbol{\theta}}(\boldsymbol{x}_{ml}|r)\right)\right] - \mathrm{KL}\left(\mathrm{q}_{\boldsymbol{\lambda}}(r)\,||\mathrm{P}(r|\boldsymbol{x})\right)\right) \tag{B.2.3}$$