Inference in vector space models of meaning

Mike Nelhams
Supervised by Dr M. Lewis

December 3, 2021

1 Introduction

A fundamental human characteristic is the ability to make inferences. For computer systems that handle natural language, the problem of Natural Language Inference (NLI) is crucial. NLI is defined as determining if a natural language hypothesis h can be reasonably inferred from another natural language premise p. An instance of such is whether the sentence ‘a skier is near the rail’ entails from the sentence: ‘a skier slides along a metal rail’ Bowman et al. (2015). In this example, human annotators described this example as entailment. Critical applications include question answering systems, text summarising and speech recognition, popular within the domains of business analytics, retail and finance Ingrid E. Fisher (2016), Veneri (2020).

Compositional semantic models attempt to deconstruct given language into smaller, more recognisable classes, such as verbs, nouns or adjectives and then they perform operations on the categorised inputs. Hyponymy is defined as one short phrase being a subtype of its hypernym phrase, e.g ‘red’ is a hyponym of ‘colour’. This paper will evaluate the effectiveness of compositional techniques and hyponymy based approaches in the context of machine learning. This paper aims to improve the accuracy of state-of-the-art NLI deep learning models, which could improve the state-of-the-art language models, leading to more responsive and more human-like systems.

2 Project plan

2.1 Research questions

This project will propose and evaluate compositional hyponymy models between popular NLI corpora, such as the Stanford NLI Bowman et al. (2015), Multi-NLI corpus Williams et al. (2018) and SciTail corpus T. Khot and P (2018). The project will focus on constructing static, dense embeddings representing hyponymy relations for each particular word using pre-trained word vector representations like GloVe Pennington et al. (2014). Following from constructing the embeddings, this project will evaluate the advantages and the limitations of integrating these embeddings into popular deep learning models such as res-nets, recurrent neural networks and transformer models. This goal can be elaborated as the following three research questions:

1. **Does compositional hyponymy offer an effective means for representing entailment between sentences in comparison to other semantic models?**
2. **How can the proposed hyponymy embeddings be used as inputs, either uniquely or in conjunction with other inputs, for deep learning language inference models?**
3. **Does using the proposed hyponymy embeddings present any benefit over state-of-the-art embeddings?**

2.2 Project timeline

Following the Gantt chart in Figure 1 as a strong guide, this project has been subdivided further into more manageable sub-tasks, in order to apply the core principles of Agile. Agile is distinct from the Waterfall approach and has been empirically shown to be more applicable than the linear Waterfall approach for smaller projects Andrei et al. (2019). Sub-tasks are assigned relative 'difficulty' ratings and each sub-task is selected from the backlog in order of importance, as long as the difficulty of the sub-task isn't too high. This ensures that the most important work is completed first and presentable progress is made each week.

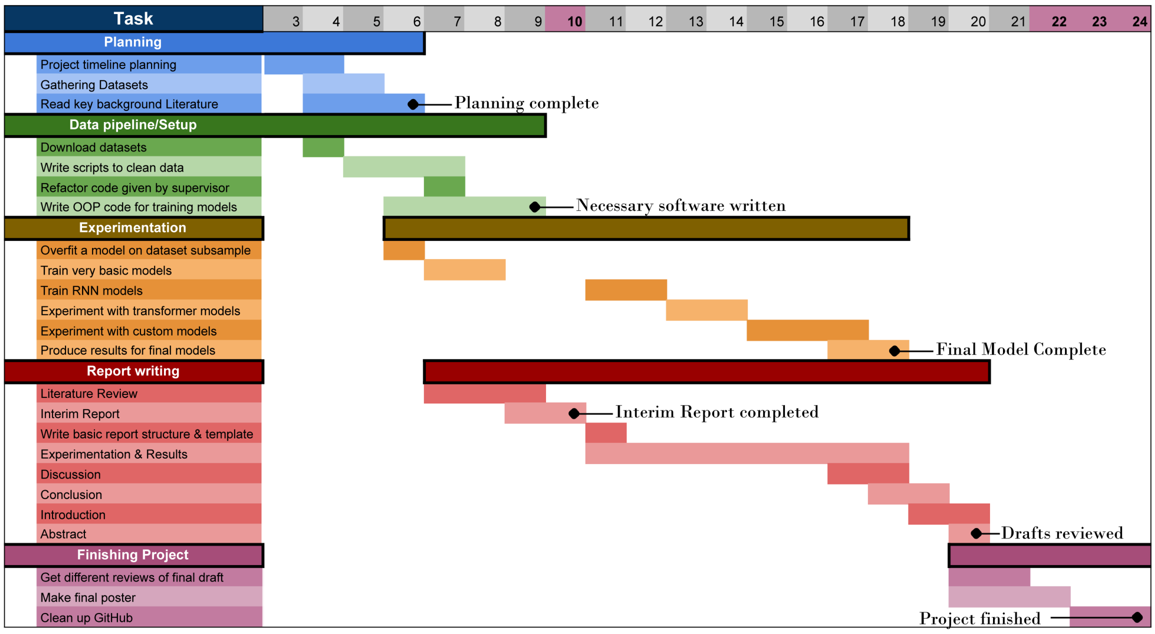

Figure 1: The Figure displays a Gantt Chart presenting the planned technical project timeline and its relevant milestones. Weeks 10 and 22-24 have been highlighted to indicate the project's assessment deadlines.

3 Literature Review

This literature review will critically summarise and compare four compositional semantic papers and the utility of the different results. The following literature review is beneficial to provide insight into the different methods available for semantics aware machine learning.

3.1 Combining symbolic and distributional models of meaning

A simplistic model developed by S. Clark and S. Pulman Clark and Pulman (2007) uses parsed sentences, where words are categorised as subjects, objects, adjectives, etc. Each lexical category is assigned an embedding vector and the sentence embedding is made by tensoring the embedding word vectors. An example is shown below for the phrase 'John likes Mary'.

$$(\overrightarrow{John} \otimes \overrightarrow{[Subject]}) \otimes \overrightarrow{likes} \otimes (\overrightarrow{Mary} \otimes \overrightarrow{[Object]}). \tag{1}$$

However, there are two major drawbacks for this technique. Firstly, the model uses a linear map of meaning, which scales spatially with order $O(d_{emb}^N)$, where d_{emb} is the size of the embedding vector and N is the number of words in a sentence, thus is very inefficient. Secondly, the model cannot compare any sentence embeddings that are not of the same lengths, which is impractical for the majority of datasets.

3.2 Mathematical Foundations for a Compositional Distributional Model of Meaning

The authors of 'Mathematical Foundations for a Compositional Distributional Model of Meaning' Coecke et al. (2010) present a rigorously defined vector space model of meaning named 'DisCoCat'. The key idea is that 'the meaning of a word can be determined by the words which appear in its contexts' and it elaborates this idea by uses parsed sentences, where words are labelled as subjects, objects, verbs, etc. in order to perform linear mapping operations. Nouns and sentences are treated as vector spaces, whereas other lexical categories such as verbs or adjectives are seen as linear maps between vector spaces. For example, adjectives are linear mappings from a noun vector space to another noun vector space.

By decomposing sentences into a series of linear mappings applied to a vector space, the spatial complexity is greatly reduced from the previous work by Clark and Pulman (2007) to only $O(d_{emb})$, however constructing the linear mappings required to transform a vector space input to a vector space output proves quite difficult. Secondly, the paper, whilst being theoretically rigorous, does not provide a complete algorithm for constructing sentence embeddings from starting input to final output. Understanding the DisCoCat model requires a very strong understanding of scientific computing, category theory and linear algebra, which makes the model difficult to implement.

3.3 Compositional hyponymy with positive operators

The paper 'Compositional Hyponymy with Positive Operators' Lewis (2019) provides an understandable, practical procedure for composing density matrices from popular word-vector representations such as Wordnet and GloVe Wordnet (2010), Pennington et al. (2014). Its approach is logical and the algorithm's results are evaluated across three single-word entailment datasets, on which it significantly outperforms other leading approaches such as HyperVec Nguyen et al. (2017). The density matrices are constructed by first collecting all the hyponyms of a given word, then summing the outer products of each hyponym's word embedding. A slightly altered example taken from the paper is shown below:

$$[\![pet]\!] = \overrightarrow{dog}\overrightarrow{dog}^T + \overrightarrow{fish}\overrightarrow{fish}^T + \overrightarrow{cat}\overrightarrow{cat}^T \ , \tag{2}$$

where $\{dog, fish, cat\}$ is the set of all known hyponyms for the word pet.

Compositional hyponymy through density matrices improves on 'DisCoCat' Coecke et al. (2010) by providing a clear, simple and rigorous implementation. Furthermore, the use of density matrices removes the necessity for parsed sentences where words are semantically labelled according to lexical categories. Words do not need to be dynamically assigned as nouns, verbs or other categories, because this information is 'contained' in the density matrices constructed from the hyponymy relations. However, a potential disadvantage of the technique, is that the spatial requirements for all words are of the order $O(d_{emb}^2)$, where d_{emb} is the length of the embeddings, therefore squaring the original embedding complexity.

3.4 Semantics-aware BERT for Language Understanding

A paper titled 'Sematics-aware BERT for Language Understanding' Zhang et al. (2020) proposes and evaluates the idea of integrating semantic understanding alongside a transformer model, in order to produce a more powerful deep learning model. It obtains 'new state-of-the-art or substantially improves results on ten reading comprehension and language inference tasks', through the addition of labelling inputs into lexical components such as objects, subjects and arguments. The other principle improvement from a standard BERT model is the inclusion of a CNN with max pooling after the final layer, which was selected due to its 'fast speed' compared to other standard neural networks, such as RNNs.

The SemBERT model achieves high accuracy without ensembling. Most leading models are ensemble learners, which combine other less-effective models, and then each sub-model predicts an output. The best output is then selected according to some metric, in order to achieve a higher accuracy. Whilst ensembles are more stable and lead to higher levels of accuracy, there are two important drawbacks. Firstly, ensemble models suffer from a gross lack of interpretability. As the number of models included from ensembling increases, the complexity of the ensemble model increases exponentially. Secondly, the computation required for ensemble learning is far greater than a single model, since it must make a prediction for each of the included sub-models. The SemBERT model also makes use of pre-trained BERT parameters, which saves on training time.

A potential criticism is the initial use of the 'PropBank' corpus for semantic role labelling. It may be unfavourable, because more modern and effective corpora exist, such as 'SemLink' Palmer (2009). However, it seems as though the authors are aware of this, because the labelling is later improved by the predictions of a secondary semantic role labeler Peters et al. (2018).

3.5 Conclusion

The paper Clark and Pulman (2007) provides a simplistic approach for constructing sentence embeddings for representing meaning, however its spatial complexity is unusable for most applications and the model is not robust enough for use in deep learning. The 'DisCoCat' model maintains the highest level of informational density and spatial complexity, yet it lacks rigorous implementation details and the iterated linear mappings may be computationally inefficient. The paper Lewis (2019) presents a simple, effective approach for producing static density matrices for given words in a sentence, yet it is uncertain whether the quadratic spatial complexity is applicable to state-of-the-art deep learning models. Lastly, the 'SemBERT' model provides a more practical implementation and it is the leading NLI deep learning model as of the year 2020 in terms of validation accuracy across all popular NLI corpora Williams et al. (2018), Bowman et al. (2015). However, all the research, except Lewis (2019), shares the limitation of requiring the input words to be labelled as subjects, nouns, etc. which is a difficult challenge itself.

If it proves feasible to use density matrices as a substitute for standard embedding vectors, such as GloVe vectors, then state-of-the-art models may see significant improvements in terms of accuracy. A core assumption is that: density matrices perform better at assigning meaning to words in the context of single-word hyponymy, therefore density matrices will contain more useful information than standard embedding vectors when used in deep learning models for NLI.

4 Progress

Following from the planned subdivision of tasks, the key tasks and their respective completion weeks are provided below in the table below. Only the key features and milestones have been included. The project code is available from the following public repository: `https://github.com/MikeMNelhams/NLI_hyponymy_analysis`.

Task	**Completion Week**	**Task**	**Completion Week**
Download readable NLI datasets	**3**	Calculate balidation loss/accuracy after each epoch	**7**
Refactor code for constructing dense matrices	**4**	Train a shallow dense neural network on SNLI	**7**
Implement full pipeline for reading NLI datasets	**4**	Implement space-efficient early stopping	**8**
Research Transformer-based implementations	**4**	Save model checkpoints during training, according to early-stopping	**8**
Research literature on different hyponymy-based approaches	**5**	Perform feature analysis on NLI datasets	**8**
Write software for training deep learning models	**5**	Represent feature analysis in graphs	**9**
Optimise existing software. ∼40% faster training	**6**	Finish writing basic literature review	**9**
Research the top-performing SNLI models	**6**	Implement f1, accuracy, recall and precision metrics	**9**
Overfit a basic model on a small dataset	**6**	Allow different data sampling techniques	**9**

4.1 Initial data analysis

The label frequencies for the SNLI dataset are provided in Figure 2. It's clear that the distributions of labels is fairly uniform across the SNLI corpus, with the unknown as an outlier, therefore the minimum expected accuracy is approximately 30%.

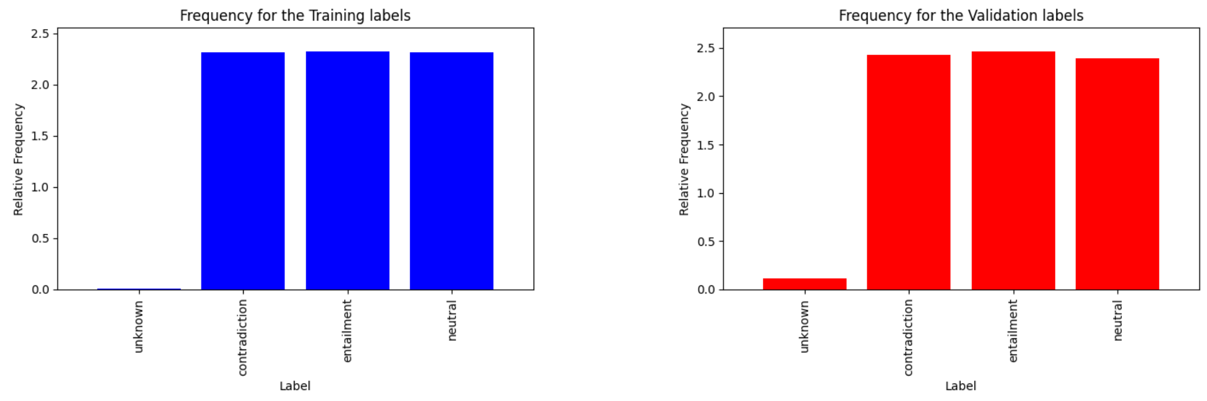

Figure 2: Illustrated above are the different label frequencies for the SNLI corpus.

4.2 Data pipeline

The SNLI corpus is the principle dataset used for testing the software, due to its simple structure. The SNLI dataset is split into training, validation and testing subsets. The datasets are cleaned by removing all punctuation, converting to lower case and removing trailing spaces.

After cleaning the datasets, the batches can be loaded either randomly or sequentially. Randomly shuffling the batches adds noise to training data, which will increase the long-run accuracy, by decreasing the ordering bias. The data is loaded in mini-batches, therefore training converges quickly to local optima. By increasing the batch size during the training, the model will converge to the training data global optimum faster than training in fixed-size batches.

The models all use pre-trained GloVe embedding vectors of embedding size 25, trained on a large dataset of Twitter tweets Pennington et al. (2014). The embedding size was selected to be as low as possible to reduce spatial complexity and increase model training times. The embedding vectors are stored as an in-memory SQlite database and all of the vocabulary that is not in the SNLI corpus is dropped from the SQlite database, to greatly reduce the lookup times.

Next, the word vectors are stacked and each sentence is padded, so that each sentence is the same length. Each provided sentence is then stacked along another tensor dimension each padded to uniform lengths. Masking is used during the model training, as to prevent repeated computation and to ensure that padding does not affect model training or testing. The model's input tensors are 4-dimensional: (batch size, number of sentences, maximum padding length, embedding length). The corresponding class labels are 'entailment', 'contradiction', 'neutral' or 'unknown' where the unknown label is where the human labelling was not unanimous.

Compositional density matrices are constructed for each of the words in the vocabulary according to algorithm described in 'Compositional Hyponymy with positive operators' Lewis (2019) and can be substituted for GloVe Vectors. Using wordnet Wordnet (2010) as a static list of all the common hyponymy relations, the density matrices for every word in the vocabulary are pre-computed. Loading the density matrices is also done through SQlite in-memory databases similar to the word vectors. Switching the inputs from word-vectors to density matrices increases the input tensor dimension shape from 4 dimensions to 5: (batch size, number of sentences, maximum padding length, embedding length, embedding length).

4.3 Training

Beginning with the criterion, the models all use categorical cross entropy to optimise the model parameters after each mini-batch. After the model has seen all of the training data, it then predicts classes for the validation set. When the model validation loss decreases or after each epoch, model parameters are saved. There are always two model parameter checkpoints saved, one corresponding to the 'most accurate' set of model parameters and the other being the most recent model. Consequently, models can be trained for a set amount of time, then halted to continue training later. If the model has consecutively not improved on the validation set predictions after a patience value, k epochs, model training will be stopped early.

The training with the default PyTorch Adam optimizer overfitting on a subset of the data without early stopping can be seen in Figure 3. The training for the entire dataset with early stopping for a 3-layer fully connected neural network can be seen in Figure 4. Both training sessions were done using standard GloVe embeddings, to serve as a baseline for all models. The curve instabilities (spikes) are due to learning rates being too large at the beginning of training.

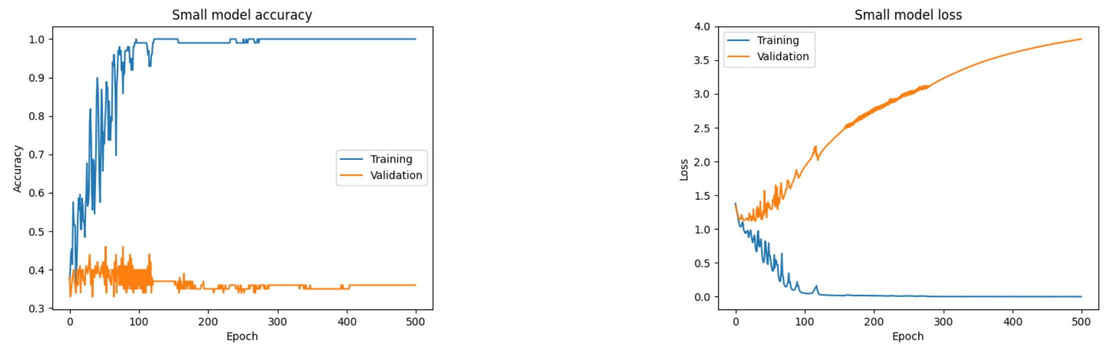

Figure 3: The graph above illustrates training to 100% over 500 epochs for 100-sample training and validation datasets. The reason the validation and training losses diverge so quickly, is due to the training dataset being unrepresentative of the sample distribution. The training to 100% proves that the training software works as intended.

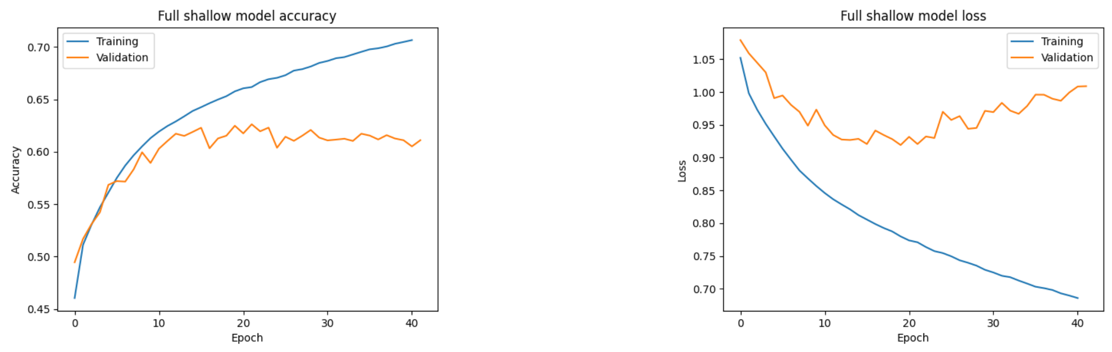

Figure 4: The graph above illustrates training with early stopping and a high patience on the entire SNLI training/validation datasets.

References

[1] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference, 2015.

[2] Mark E. Hughes Ingrid E. Fisher, Margaret R. Garnsey. Natural language processing in accounting, auditing and finance: A synthesis of the literature with a roadmap for future research. *Intelligent Systems in Accounting, Finance and Management*, 23(3):157–214, 2016.

[3] Giacomo Veneri. Natural language processing tutorial with sota 2020 python packages [online]. Available from: `http://jugsi.blogspot.com/2020/05/natural-language-processing-tutorial.html`, 2020. Last Accessed: 29/11/2021.

[4] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference, 2018.

[5] A. Sabharwal T. Khot and Clark P. Scitail: A textual entailment dataset from science question answering [online]. Available from: `http://ai2-website.s3.amazonaws.com/publications/scitail-aaai-2018_cameraready.pdf`, 2018. Last Accessed: 30/11/2021.

[6] 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`.

[7] Bogdan-Alexandru Andrei, Andrei-Cosmin Casu-Pop, Sorin-Catalin Gheorghe, and Costin-Anton Boiangiu. A study on using waterfall and agile methods in software project management. *Journal Of Information Systems & Operations Management*, pages 125–135, 2019.

[8] Stephen Clark and Stephen Pulman. Combining symbolic and distributional models of meaning, 2007.

[9] Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. Mathematical foundations for a compositional distributional model of meaning, 2010.

[10] Martha Lewis. Compositional hyponymy with positive operators. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 638–647, 2019.

[11] Wordnet. About wordnet. Available from: `https://wordnet.princeton.edu/citing-wordnet`, 2010. Last Accessed: 30/11/2021.

[12] Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde, and Ngoc Thang Vu. Hierarchical embeddings for hypernymy detection and directionality, 2017.

[13] Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. Semantics-aware bert for language understanding, 2020.

[14] Martha Palmer. Semlink: Linking propbank, verbnet and framenet. In *Proceedings of the generative lexicon conference*, pages 9–15. GenLex-09, Pisa, Italy, 2009.

[15] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.