

Department of Computer Science



Submitted in part fulfilment for the degree of BEng.

Do we give general language models a chance?

Analysing State of the Art NLP Benchmarks for Grammatical Distribution Drift

Adam Hawley

23 March 2021

Supervisor: Simos Gerasimou

Contents

Executive Summary	v
1 Introduction	1
1.1 Aims	1
1.2 Restrictions	1
1.3 Motivation	1
2 Background	4
2.1 Encoding Text	4
2.2 A Brief History of Language Models in Machine Learning . .	5
2.3 Transformers	5
2.4 Benchmarks	6
2.5 Embeddings vs. Parse Trees	6
2.6 Data Drift	7
2.7 Popular Pre-Training Datasets	7
2.8 Dataset Splitting	8
3 Methodology	9
3.1 Overview	9
3.2 Data Representations	10
3.3 Benchmark Analysis	13
3.4 Investigating Detected Drift	15
3.5 Implementation	15
4 Results	17
4.1 Benchmark Analysis	17
4.2 Investigating Detected Drift	21
4.3 Discussion	23
4.4 Dependency Analysis	26
5 Project Conclusion	28
Future Work	28
Appendices	29
A Appendix	30

List of Tables

4.1	Proportion of POS-trees in test split not present in train splits.	17
4.2	Proportion differences of POS-trees between train and test sets.	18
4.3	Dependency labels with drift.	20
4.4	Hypothesis testing for shuffled splits.	23
5.1	GLUE and SuperGLUE tasks used during the project. . . .	30
5.2	List of syntactic dependency labels used by the SpaCy EN Transformer Model.	30

List of Figures

3.1	Visualisation of the parse-tree of the phrase 'Here is an example tree.'	10
3.2	Workflow for use of both grammatical encodings.	16
4.1	Snippet from the EvidentlyAI dashboard visualisation distribution difference in SST-2 train and test splits.	21
4.2	Distribution of number of dependency labels calculated as showing grammatical distribution drift between random splits by task (Left: GLUE; Right: SuperGLUE)	22
4.3	GLUE: Out of distribution syntactic dependency labels in official task splits.	27
5.1	Box and whisker plot of a subset of SuperGLUE tasks.	32
5.2	Box and whisker plot of a subset of GLUE tasks.	32

Executive Summary

This project aims to present a novel methodology for identifying grammatical distribution drift. To showcase the methodology, two major natural language processing (NLP) general language model benchmarks are tested for grammatical distribution drift between train and test splits.

In a recent publication, state of the art (SoTA) language models have been shown to have grammatical pitfalls with high failure rates on specific grammatical constructs. At the time of writing, there is also a growing amount of academic interest in data quality and engineering of machine learning systems. This project is based on these two motivations. It aims to provide a new perspective on feature engineering in NLP machine learning. Potentially this method and relevant grammatical encodings could be used to investigate the causes for the aforementioned pitfalls. However due to the size of SoTA pre-training datasets and models, this particular investigation remains out of the scope of this project. Due to computational restrictions, the use of benchmarks rather than (pre-)training datasets is adopted.

This project presents two new encodings of grammatical attributes to be used in conjunction with current SoTA benchmarks and datasets. These encodings are designed to be used for a method for detecting grammatical distribution drift which is also presented in this report. The method allows a user to investigate the degree to which two datasets vary in grammatical distribution by detecting a difference in distribution in the frequency of each syntactic dependency for each textual attribute of a data point. Following this, the method expands to be able to determine if a given grammatical drift is the cause of random shuffling or if the splits are statistically likely to have been (intentionally or unintentionally) manufactured to contain a difference in distribution.

The two grammatical encodings presented include two representations. Firstly, the POS-tree which is a tree representation in which the nodes are made up of part-of-speech tags which have been inferred from tokens in the text. The second representation is the Syntactic Dependency Counter (SDC) which extracts the syntactic dependencies from text and stores the frequency of each category of dependency.

The first step of the method encodes the textual features of data points into their relevant grammatical encoding. Then using the SDC encoding, the frequencies of syntactic dependency labels are treated as features of each data

Executive Summary

point. By considering the distributions of each syntactic dependency in the two datasets, I am able to determine if the two distributions are statistically likely to have been drawn from the same distribution. Hence I am able to determine if there is a difference in distribution for each syntactic dependency label and subsequently evaluate the total difference in grammatical distribution.

For the second stage of the method, in order to provide insight into the construction of the two datasets, I perform a repeated merge, shuffle and resplit of the data. This process allows me to statistically determine if the original configuration of the data is substantially likely to have been created in a way that methodically produced a difference in distribution or if the data is likely the product of random shuffling.

As mentioned above, to showcase the method, this project analyses 2 popular SoTA NLP benchmarks, GLUE and SuperGLUE, for a difference in grammatical distribution between task train and test splits. Using the presented method exposes grammatical drift in all but one task. These results also include identifying intentional use of out-of-distribution data exclusively in test splits. Powerfully, specific to the Corpus of Linguistic Acceptability (CoLA) test, the results show that, through supervision of the grammatical structures used in datasets, out-of-distribution data can still be used to test models on a more representative set of data without possibly disadvantaging the model by introducing severe grammatical distribution drift.

The work done in this project does not present any particular legal, social, ethical, professional or commercial issues.

1 Introduction

1.1 Aims

In this project, I aim to present a methodical approach to measuring grammatical distributions in NLP datasets. I use this method to investigate the grammatical distributions in train/test splits of popular NLP benchmarks. Along with the method itself, this report presents and evaluates two forms of grammatical encoding which are used in combination with the method.

1.2 Restrictions

Even though this report analyses state of the art benchmarks and discusses datasets used to train and test state of the art models, the sizes of the models themselves and the datasets required to train and test such large models mean that it has become increasingly common to require extremely large amounts of computational power [1]. The lack of availability of such high levels of computational power means that during this project, several compromises are made to achieve satisfactory results. Namely, the scope of the project has adjusted over time, there remain two sections of experimentation which would supplement this project well but remain untested for future work. Firstly, an analysis of how drift in grammatical distribution affects state of the art (SoTA) model performance could support greatly the use of the findings within this project. Following this, an analysis of the grammatical distribution of pre-training datasets rather than fine-tuning datasets would help indicate to what extent the models have already 'learned from' other grammatical distributions.

1.3 Motivation

1.3.1 Testing NLP Models

This project was greatly inspired by Checklist [2]. Checklist presents a framework to create and run behavioural tests of natural language processing (NLP) neural network models. In the relevant paper, the authors report

1 Introduction

failure rates of up to 100% for state of the art models such as BERT-base and the models provided via a paid API by Google, Microsoft and Amazon against test cases which seem very straightforward to any English speaker.

It is clear that it will not be acceptable for production-level models which are used in industry to perform so poorly on realistic and simple examples such as the failures described above, see Example 1.1 for an example from [2]. Diagnosing why models do not perform given these different examples is not an aim of Checklist. Checklist does not offer any explanation or suggestion for a given model's performance.

Explainability and clarity of AI is a hot topic in machine learning [3]–[6]. Particularly in the domain of NLP, where researchers and businesses are seeking valid explanations for the ethical issues I go on to mention in Section 1.3.2. The Transformer architecture is present in most SoTA models in NLP and visualisation of attention [7] provides some clarity into neural networks which are usually seen as black boxes. However, if there are blatant issues at that stage in the pipeline then there is still little diagnosis that can be done to provide direction for the programmer to investigate. Too often, much time is lost investigating architectures, implementation details and model configuration when the problem lies in the earliest part of the pipeline: the data.

1.3.2 The Importance of Data in NLP Neural Networks

It is undisputed that data quality for machine learning is of the utmost importance [8] and yet the focus on creating performant models often falls on model architecture and fine-tuning rather than on the quality of the data [9]. However, in recent years attention around data quality has started to grow [9]–[12].

One reason for the growth in interest around data in machine learning pipelines is due to the pressure researchers and engineers have been under to explain why models are perpetuating social inequality through echoing stereotypes [13]–[15], under-representing social groups [16]–[18] and even creating negative biases towards mentions of disability [19].

1.3.3 Data Quality

As discussed in Section 1.3.2, data engineering and analysis is often a disproportionately overlooked part of machine learning. Due to the variance of data quality, structure and format, it is impossible to define a step-by-step guide on processing data for every machine learning task. There are many known issues that can arise from errors in data including datasets with skewed

1 Introduction

distributions, mislabelled data and training a model which works on training and test data but not real-world data.

One difficulty with verifying the quality of data is that it is labour-intensive. It can be difficult to perform an automatic or computed analysis of the quality of data since often you will be looking for problems that you do not know exist yet. Without insuring the data is of sufficient quality, how can you know that any lack of performance is not caused by a problem in the data? There can be many issues in data that lead to poor/misleading performance in machine learning models.

With certain problems, recording data can be expensive [20], difficult or even dangerous [21]. Equally, some problems may simply have extremely complicated relationships which need to be learned. For models which learn through gradient descent and gradual weight adjustment, large amounts of data will be required to acquire this complicated knowledge. For example, the pre-training of GPT-3, a headline-worthy [22] pre-trained general language model, was done on over 260 billion tokens [23].

It is essential that training data represents the data in the real world. In NLP, out-of-domain data problems make up their own extensive area of research [24]. This project focuses specifically on identifying non-representative training data but where research in this topic usually focuses on semantics and domain-specific data; this project considers the linguistic properties of data such as grammatical structures used and sentiment-in-sentence distribution.

For example, English speakers are used to structures which Checklist has been used to show state of the art models fail consistently on [2], see Example 1.1 from [2]. A sentence with a negated positive phrase split by a clause of neutral sentiment should remain negative. During the behavioural tests done by Ribeiro et al., Microsoft's, Google's and Amazon's commercial Sentiment analysis models followed by BERT and RoBERTa had failure rates of 98.4, 100.0, 100.0, 74.0 and 30.2 respectively on tests with this structure [2].

Example 1.1. 'I wouldn't say, given it's a Tuesday, that this pilot was great'.

This project presents a way to use extended sentence parse trees and syntactic dependency labels to compare and contrast training and test datasets of different benchmark tasks to investigate train-test grammatical distribution disparity.

2 Background

2.1 Encoding Text

Embeddings Neural networks work using numerical inputs therefore text inputs need to be encoded into a numerical representation. Usually, this is done with word embeddings or encodings [25], [26] which are vector representations of words. Variations of word embeddings include sentence embeddings [27] as well as one-hot encodings or, similarly, count-vectorising.

Part-of-Speech Tagging Part-of-Speech (POS) tags provide a way of labelling tokens in text with a grammatical class. A basic form of POS-tagging is often taught in schools with a subset of POS-tags usually including but not limited to the following example tags: 'verb', 'noun', 'connective', 'adjective', 'adverb' etc. POS-tagging is one of the attributes I go on to use in combination with parse trees to classify sentence structures.

The action of tagging was originally a completely manual task but has more recently become a task completed using a variety of methods. These methods include dynamic programming algorithms (such as the popular Viterbi algorithm [28]). It is common to see hidden Markov models implemented in POS-tagging algorithms [29]. As with many modern NLP elements, machine learning is now very popular for POS-tagging. In this case, computers analyse corpora of text which has been appropriately tagged already and try to learn how to tag tokens through supervised learning [30], [31].

Syntactic Dependency Parsing The process of parsing syntactic dependencies involves identifying how separate tokens depend on others as a result of grammatical/syntactic rules. For example, given the phrase 'He ate cake.', 'he' has a subject-dependency on the verb 'ate' due to the subject-verb relationship.

Named Entity Recognition Named entity recognition involves labelling the named entities within a given piece of text. Commonly named entities include people, geo-political entities (e.g. states or countries) and organisations such as businesses, charities etc.

2.2 A Brief History of Language Models in Machine Learning

Initially, vector representations of words were learned [25], [26] and then used in networks designed to only answer one task. This means that once a model was trained, the learned features could not be reused on a different task. Recursive neural networks (RNNs) [32]–[40] and subsequently particularly long-short term memory networks (LSTMs) [41]–[44] were used to improve representations by keeping track of the context around words using memory built into the networks and the recursive nature of the networks [45]. However, these still revolved around task-specific networks.

Following those approaches, pre-training models (especially models with Transformer architectures) began to grow in popularity [23], [27], [46]–[52]. The concept of pre-training can be seen as a form of transfer learning [53]. Pre-training often consists of unsupervised learning tasks such as predicting missing words in a given sentence [23], [48]. After being pre-trained, these models are then trained on task-specific training datasets so that the model can learn how to use the ‘understanding’ it has gained from the pre-training step to solve the task-specific problems. This step is usually referred to as fine-tuning the model. Models which have been pre-trained and are task-agnostic are often referred to as ‘general language models’.

2.3 Transformers

As mentioned above, the current state of the art models generally use the Transformer architecture. The Transformer architecture was introduced by Vaswani et al. in 2017 [46]. At the time, attention mechanisms were proving successful additions to recursive and encoder-decoder models [54]–[56]. The Transformer was the first entire architecture to be oriented around an attention mechanism.

Attention mechanisms in NLP refer to a method of weighting the importance of other inputs during a sequential input. For example, standard recursive neural networks work by passing the output of a given input back into the model along with the next input. Attention mechanisms provide the model with a way of learning specifically which other inputs to ‘pay attention to’. For example, given the input ‘I am really interested in cars’, when the model is considering the input token ‘interested’, it may ‘pay attention to’ the words ‘cars’ and ‘I’ in order to understand that the subject of the sentence has a positive sentiment towards cars.

In this project, I make use of RoBERTa [49] as a key part of my grammatical encoding pipeline. At the time of writing, the base configuration of RoBERTa

still holds 10th place in the SuperGLUE leaderboard and furthermore, 4 out of the top 10 models on the leaderboard have used different versions of RoBERTa to achieve their scores¹. RoBERTa improves on BERT by altering the pre-training of BERT with a focus on the importance of a variety of different hyperparameters and also find that BERT’s performance could be greatly improved by increasing the amount of pre-training performed [49]. Its predecessor, BERT uses Bi-directional Encoder Representations from Transformers (hence the name) which means that during its pre-training stage, it learns to depend on both the left and right context of a token and therefore considers the tokens to the left and to the right of the token input when encoding the token.

2.4 Benchmarks

GLUE [57] and SuperGLUE [58] are both benchmarks for general-purpose language understanding systems. They consist of several subtasks for which each model is evaluated. Therefore in order to score highly in the benchmark, a model has to perform well on a variety of tasks. GLUE contains nine Natural Language Understanding (NLU) tasks. In this project, I use several to analyse how performance differs between models trained on different datasets.

One such task is the Binary Classification Stanford Sentiment Treebank Benchmark (SST-2) [59]. As a benchmark, the task is a binary classification sentiment analysis task using the movie reviews dataset created by Pang and Lee [60] and parsed by the Stanford Parser [61]. In the binary benchmark, each sentence is marked as either a positive statement or a negative statement. However, in the fine-grained task each sentence is labelled as one of 5 categories: ‘very negative’, ‘negative’, ‘neutral’, ‘positive’ and ‘very positive’. Later in this report, I address the methods and aims addressed in SST-2 (see Section 2.5). For a full list of subtasks analysed in this project see Table 5.1.

2.5 Embeddings vs. Parse Trees

Word-embedding representations are used to represent text in many SoTA machine learning use cases [23], [51]. Sentence embeddings are also used to provide vector representations of sentence semantics [27]. However these embeddings capture a word or sentence’s semantic meaning alone and therefore lose attributes such as syntactic dependencies [25].

¹<https://super.gluebenchmark.com/leaderboard>

Parse trees are a common representation of text used in NLP [31]. The Stanford Sentiment Treebank (SST) [59] is one of several tasks which make up the GLUE benchmark. SST uses a treebank containing parse trees of 11,855 sentences. The aim of the paper and the treebank is to create deep neural network models with higher accuracy on sentiment analysis tasks by training those models on the trees. Socher et al. introduce Recursive Neural Tensor Networks which represent phrases as trees where leaf nodes are labelled with tokens based on grammatical dependencies and all nodes have an inferred value of sentiment. The sentiment value is first derived from the leaf nodes and then combined as the tree is traversed from the leaf nodes upwards. This project adapts the Recursive Neural Tensor Network models for use with any attribute such as POS-tags.

2.6 Data Drift

The concept of data drift is usually used to describe the effect of a distribution change between training and deployment data [62]. Popular data drift detection libraries such as Alibi-Detect [63] conduct data drift detection on NLP datasets however, they work by first encoding the text. By using the word embedding representations of the text and throwing away the explicit grammatical properties which can be inferred from the raw text I believe that they could be ignoring possible drift present in the distribution of grammatical structures used. To contrast this, in this project I test specifically the grammatical attributes of text for distribution drift between training and test splits.

2.7 Popular Pre-Training Datasets

BookCorpus The BookCorpus was created to train a sentence embedding model which was part of the work done by Zhu et al. to capitalise on the manual transformation that has been done to make films out of existing books. The dataset itself consists of books written by unpublished authors with a length of more than 20k words. BookCorpus [64] has since popular dataset used to pre-train the likes of DeBERTa [51], BERT [48] and other BERT-based models.

English Wikipedia Dump Wikimedia provide dumps of all Wikimedia wikis with splits for each language. These dumps are provided at least as frequently as each month so continue to grow in size. Models which have used these dumps for pre-training include BERT [48] and T5 [65].

CommonCrawl Amazon Public Datasets store the CommonCrawl dataset. CommonCrawl is a publicly available corpus of web crawl data and has been used to train the likes of GPT-3 [23]. A subset of CommonCrawl, STORIES, was created and used by Trinh & Le [66] to show that diversity of training data plays an important role in test performance. Further subsets of CommonCrawl are also used in the training of non-English models such as PANGU- α [50]. STORIES was also used in the pre-training of both RoBERTa [49] and DeBERTa [51].

Open WebText Due to issues in the quality of the entire CommonCrawl dataset [66], GPT-2 was pre-trained on a new corpus of web crawl data created just for GPT-2. The new corpus came from scraping web pages that had already been curated/filtered by humans. The pages to crawl came from following all outbound links from Reddit on posts that had at least 3 karma. Open WebText is a recreation of this dataset and has subsequently been used to pre-train RoBERTa and DeBERTa [51] a new corpus of web crawl data created just for GPT-2. The new corpus came from scraping web pages that had already been curated/filtered by humans. The pages to crawl came from following all outbound links from Reddit on posts that had at least 3 karma.

2.8 Dataset Splitting

There are many ways to split a dataset into train and test splits. This project uses statistical methods to analyse the probability that the grammatical distributions of datasets have been split randomly.

One simple method is to shuffle the data and then randomly split or to randomly sample the data as is used in [67]. Even though this approach is simple, some cases call for greater control over the splits. For example, stratified sampling is commonly used to ensure that each category of a data point in a train split is suitably in the test split [68]. Furthermore, commonly datasets are adversely split by reserving data from other distributions to be used in the datasets or by exploiting common weaknesses [69]. This can be done to increase the difficulty of a benchmark intentionally [70].

Alternatively, one can opt to use the ‘hold x out’ strategy in which every x^{th} data point is ‘held out’ and reserved for the test split [71].

It is important to note that even though a wide variety of dataset splitting techniques can be used, any kind of intentional proportional enforcement is usually to control the distribution of data point labels and not secondary attributes such as grammatical distribution [68].

3 Methodology

3.1 Overview

The aim of this project is to explore the presence of grammatical distribution drift in NLP benchmark tasks. Below I give an overview of my approach with details on the following pages.

Data Representation

In order to compare the splits of benchmarks I need a way to encode the grammatical attributes of data points. In the first section of my methodology, I outline a couple of options I considered, how I constructed them, how they will be parsed from the textual attributes of the data points and finally an evaluation. I outline the different positives and negatives of each representation in general. Section 3.2 explains fully the data representations created and how they were evaluated.

Benchmarks

With established data-types implemented to track the grammatical distributions of datasets I can begin to test the datasets themselves. During the benchmark investigation portion of the paper I run my grammatical analysis on the test-train splits of different benchmarks. See Section 3.3 for the detailed methodology of my benchmark analysis.

Causal Analysis

To complement the analysis performed on the benchmarks I perform a further investigation to try and provide answers and explanations for the results found in the benchmark analysis. One way I do this is by using random shuffling of the test and train data to analyse the probability that the official split was the result of random shuffling.

Linguistic Attributes

Due to the extensive use of syntactic dependencies, the final part of this project consists of a small analysis of the frequency of syntactic dependency labels to see if any relationship between complexity and grammatical distribution drift is present.

3.2 Data Representations

3.2.1 POS-Trees

In order to compare datasets, I knew I was going to have to analyse certain features of the datasets. To keep track, compare and manipulate these features I had to create a model for storing this information. Parse-trees are frequently used to represent grammatical structures in text including in the SST-2 benchmark as described in Section 2.5. Using parse-trees enables me to store multiple attributes about each token without ever losing the structure of the input.

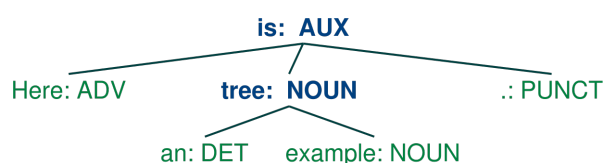


Figure 3.1: Visualisation of the parse-tree of the phrase 'Here is an example tree.'

Parsing Text To go from text to trees, I made use of two very popular and famous NLP python libraries called SpaCy¹ and NLTK². My implementation, `SentTree`, is attribute agnostic in that nodes of the tree do not have to have any grammatical representation. For example, as with the trees in the Stanford Treebank [59], the trees could be used to represent sentiment trees where leaf nodes represent the sentiment of tokens in a sentence. However, due to the grammatical focus of this project, when I refer to parse-trees or trees in general I am referring to single attribute POS trees unless otherwise specified.

In order to parse a sentence into the tree representation, firstly the sentence is passed through a SpaCy NLP pipeline. I made use of SpaCy's pre-trained

¹<https://spacy.io/>

²<http://www.nltk.org/>

3 Methodology

general language Transformer model³ which have many pipeline steps implemented including POS-tagging, syntactic dependency parsing, named entity recognition and lemmatisation. However, for parsing text into POS-tree representations, all steps except for dependency parsing and POS-tagging were removed to increase performance. The SpaCy Transformer model uses the base configuration of RoBERTa. The sentiment attribute of each token can be analysed with the use of an extension called spaCyTextBlob⁴ (integrating TextBlob⁵ into the spaCy pipeline).

Even though visualising the trees themselves does not play a large role during the analysis stage of the project, it was essential for quality assurance; ensuring that the Transformer model provided by SpaCy and the tree-builder worked correctly together. See Figure 3.1 for an example parse tree representation.

Measuring Tree Frequencies For each data point in each dataset, I split the text into sentences and then parse each one into a tree representation. By hashing each instance of each tree, I count its appearances in the text.

When comparing the counting results of two datasets, I iterate through the trees in each set. I then calculate what percentage of the results are made up by the tree. The comparison then also iterates through each dataset and records the difference in distributions. For example, if 4 sentences which appear in dataset 1, d_1 , share the same abstract-tree representation, t , out of 20 sentences in d_1 then tree t is said to represent $\frac{1}{4}$ of the sentences in d_1 .

3.2.2 Syntactic Dependency Counter (SDC)

To complement my investigation using POS-tags, I also utilise the syntactic dependency parsing stage to count the appearance of certain relationships within text spans seen by models. Namely, I created a syntactic dependency counter which is added to each data point in a given dataset.

Parsing Text For tasks with one input, I run that input through the same SpaCy pipeline which is used for POS-tree parsing and count the number of occurrences of each syntactic dependency label. For a full list of labels used in this process, see Table 5.2. I then compare these counts between the train and test splits to verify that there is no drift in grammatical distribution between splits. For tasks with multiple inputs, the process is very similar however, each textual input is analysed separately.

³https://spacy.io/models/en#en_core_web_trf

⁴<https://spacytextblob.netlify.app/>

⁵<https://textblob.readthedocs.io/>

For example, the CommitmentBank [72] is a 3-class entailment classification task in SuperGLUE which was created to investigate the projection of finite clausal complements. This task provides the model with a premise, a stream of text which includes at least one embedded clause and a hypothesis. The model is then asked to classify the hypothesis as entailing or contradicting premise or a final third option which the model can use if the hypothesis neither entails nor contradicts the premise. For this task, my analyser looks at each data point and counts the number of appearances of all syntactic dependencies within the premise, then the stream of text and finally, the hypothesis and each distribution of frequencies remain separate. These frequencies are recorded before performing the same action on the hypotheses.

3.2.3 Representation Evaluation

Initially, the POS-tree representation seems like a strong candidate for a representation to use in this project for several reasons. My implementation can handle multiple attributes therefore analysis can be performed across any token-attribute such as sentiment, syntactic-dependency or word embedding. Furthermore, the representation comes with a very convenient and intuitive visual representation, this means that complicated structures can be quickly explained by looking at the tree which can display the token and the attribute side by side in the correct position on the tree. On the other hand, the syntactic-dependency counter (SDC) representation can be used in conjunction with a graphing library to visualise one data point with respect to other data points but has no intuitive visualisation of a single point alone. This means that the SDC representation makes single data points harder to visualise based on counts alone as it would require a complete understanding of all possible syntactic dependencies and how they commonly link.

Example 3.1. Equal SDC values:

My dog is the winner. (AUX (NOUN PRON) (NOUN DET) PUNCT)

The winner is my dog. (AUX (NOUN DET) (NOUN PRON) PUNCT)

However, due to the fine-grained nature of the POS-trees it means that naturally, the groups of POS-trees have much lower frequencies. This results in a couple of downsides. Firstly, the tree representation places importance on the position of subtrees within a given tree and therefore does not bucket together two trees with the same grammatical features but in different places in the hierarchy. For example, given the sentences in Example 3.1, these two sentences share the same SDC value as they contain the same syntactic dependencies but because of the change in order, they are represented by different POS-trees. Even though this positional dimension can add extra information and a greater understanding of the distributions (especially

3 Methodology

important when analysing models which do not use bi-directional encoding so position matters even more), it results in low frequencies as it is much easier for samples to fall into different tree representations. The low frequencies also make it much harder to statistically compare distributions for drift. Since this requirement is so strict, it could be useful if someone was trying to cleanse a dataset so that it was limited to a specific set of grammatical structures. However, in this project, I test datasets from tasks that are not required to have any grammatical limitation by design.

The granularity of POS-trees also has a negative bi-product on its scaled graphing interpretations. Because of the low frequencies it means that graphing an entire dataset of POS-trees causes extremely dense histograms with the majority of POS-tree counts being at 1 or 2. Potentially this issue could be overcome when analysing much larger datasets such as pre-training datasets as the higher number of data points may begin to compensate for the granularity of the representation. Unfortunately, that analysis would require more computational power than was available during the course of this project so it remains open.

Finally, there is a key distinction to be made between my implementation of the two encodings. The POS-tree can be extracted from only one whole sentence at a time while the SDC representation does not have such a limitation. This could benefit as it has a weaker assumption about the text being processed but also a weakness as it could hinder comparisons between data points.

Considering all of the points raised above, I use both representations during the project. For statistical analysis, I use the SDC encoding due to its similarity to standard numerical data point features. While I present some results using the POS-Tree, I also later suggest some of the practical use cases to which it may lend itself.

3.3 Benchmark Analysis

Due to the importance placed on benchmarks to distinguish models based on their performance, given the ability to analyse syntactic distributions gave me the perfect tool to analyse an element of these benchmarks which is often not explicitly explored. During this part of the project, I analyse train/test splits of benchmark datasets to see how much they differ.

3.3.1 Benchmark Choice

As outlined in Section 2.4, GLUE and SuperGLUE are popular benchmarks used to assess general language models' level of natural language understand-

ing. The benchmarks have been used to evaluate and compare state of the art models such as GPT-3 [23], Google's T5 [65], DeBERTa [51] as well as many others. Therefore, it is due to this popularity and regular appearance in papers presenting state of the art models, that it makes sense to test the tasks included in the two benchmarks. See Table 5.1 for a full list of tasks analysed in this project.

3.3.2 POS-Tree Counting

Despite the majority of the statistical analysis of this project being performed on the SDC representations of datasets, initially, I capitalise on the granularity of the POS-tree to check for disparity between the train and test splits. The POS-tree encoding stage produces a dictionary mapping with string encodings of the POS-trees as keys and sentences which have been resolved to the POS-tree as the values. These missing trees are found by iterating through the POS-tree representations of the data points in the test splits and then querying the train split's dictionary with the string representation of the POS-tree. If the query returns with a set of sentences then it is given that that specific type of POS-tree has been seen by models in the train split at least once. If not, then the tree is completely new and has not been seen in the training set.

Following the use of POS-trees for finding missing trees, I use the tree counts which have been found during the POS-tree encoding of the datasets to see if the proportion of specific POS-trees vary between train and test splits.

3.3.3 Syntactic Dependency Label Drift

Each task in the GLUE and SuperGLUE benchmarks provide the model being tested with one or more textual features. For every feature, I encode the frequency of each of the 45 different syntactic dependency labels. This encoding is performed on each data point in both the train and test splits of the task. Following the encoding, on a label by label basis, I perform a statistical test on the frequencies from both splits to determine if the frequencies of the label in both splits are statistically likely to have been drawn from the same distribution.

3.3.4 Conclusion

By using the methodology outlined in this section I hope to gain a firm understanding of which tasks, if any, present a difference in distribution between train/test splits. This information allows me to continue the investigation

to test if a drift in distribution affects the performance of models on such subtasks.

3.4 Investigating Detected Drift

I subsample the tasks based on the results from the initial benchmark analysis. Using this subsample, I analyse the datasets further including by iterating through alternative permutations of the datasets to see investigate how alternative splits could have been made.

3.4.1 Recreating Other Splits

In order to indicate whether a difference in grammatical distribution has been introduced to different splits for certain tasks I follow a simple procedure in order to find a distribution of feature drift. After recording the initial grammatical distribution drift between splits, I merge the splits before shuffling the data and creating random splits. The new random splits are analysed for grammatical distribution drift and recorded before repeating many times. By repeating this step many times I am able to estimate a distribution of feature drift and calculate the probability of the initial drift appearing as a result of random shuffling.

3.4.2 Statistical Analysis

During this part of the project, I perform a statistical analysis on the different splits. In Section 3.2 I explain the difference in granularity between the two suggested representations. To complement the statistics gained following the method outlined, I also sample and analyse the BookCorpus [64] which contains data from a single source and is not processed into multiple splits.

Even though it would be optimal to use the entire datasets, the datasets are extremely large. As discussed in Section 2.7, BookCorpus is a popular pre-training dataset that consists of sentences from unpublished books. I shuffle samples of the dataset together before performing the same procedure of recreating splits on the shuffled data.

3.5 Implementation

It is key that the work produced during this project is reproducible and maintainable. To enable this, I used popular, free and platform-agnostic

3 Methodology

open source projects which are listed below. Specifically, HuggingFace and SpaCy are frequently used in SoTA NLP research [73]–[75] due to their implementations of SoTA methodology such as the parsing pipelines used in this project. I have also included a flowchart that summarises the methodology and the stages at which each library is used.

Tools and languages:

- All steps of the process are implemented in Python 3.8.
- Datasets provided using HuggingFace⁶.
- Grammatical encoding implemented with SpaCy and NLTK (exclusively for POS-trees)
- Statistical analysis completed using Pandas⁷, SciPy⁸
- EvidentlyAI⁹ and seaborn¹⁰ for data visualisation.

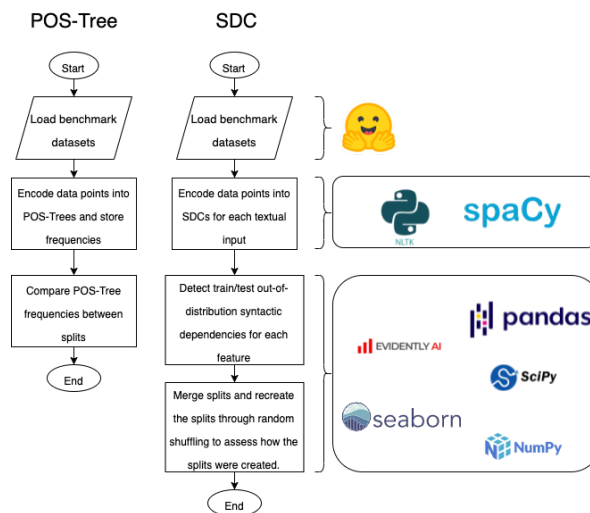


Figure 3.2: Workflow for use of both grammatical encodings.

3.5.1 Conclusion

As a result of this stage of the project I aim to have provided further insight and explanation for the results found in the initial benchmark analysis section. Using the method above I hope to analyse specific datasets to investigate whether it is possible that data from a separate distribution has been used to create test splits.

⁶<https://huggingface.co/>

⁷<https://pandas.pydata.org/>

⁸<https://www.scipy.org/>

⁹<https://evidentlyai.com/>

¹⁰<https://seaborn.pydata.org/>

4 Results

4.1 Benchmark Analysis

4.1.1 Missing Trees

As discussed in Section 3.3.2, I use the granularity of the POS-trees to see if there are any POS-trees that are used in the test splits of tasks that were not encountered within the train split. Table 4.1 shows a breakdown of the proportion of the POS-trees in the test split which will not be seen by a model while training on the train split.

Table 4.1: Proportion of POS-trees in test split not present in train splits.

Benchmark	Task	Proportion of Test Set Missing
GLUE	QQP	13%
	COLA	23%
	WNLI	31%
	STSB	33%
	QNLI	43%
	RTE	65%
	SST	80%
	MRPC	89%
SuperGLUE	MultiRC	4%
	COPA	25%
	WSC	27%
	WiC	35%
	BoolQ	58%
	ReCoRD	67%
	RTE	80%
	CB	82%

4.1.2 Initial Drift Investigation

POS-Tree Comparisons

When comparing the POS-tree counts of train and test splits, for each tree I assign a share value for the two splits. This share value represents the proportion of the split that were encoded to the given POS-tree. Then I find the difference between the two share values to represent the shift in proportional representation. In Table 4.2 I show the sum of these differences on a task-by-task basis.

Table 4.2: Proportion differences of POS-trees between train and test sets.

Benchmark	Task	Total Tree Share Diff (2 dp.)
GLUE	QQP	0.36
	COLA	0.58
	STSB	0.66
	QNLI	0.81
	RTE	0.81
	MRPC	0.95
	WNLI	0.98
	SST	1.23
SuperGLUE	COPA	0.56
	CB	0.89
	WiC	0.60
	BoolQ	0.79
	RTE	0.93
	MultiRC	0.95
	ReCoRD	0.96
	WSC	0.98

As discussed in the representation evaluation (Section 3.2.3), due to the low frequencies caused by the granularity of the POS-tree representation, their frequencies are not conducive to traditional statistical analysis. Hence, I make no statistical conclusions based on the tree share difference values. However I have included them here to show the relationship between tree share difference and the statistics found from the SDC representation and to show how the difference in representation affects the statistics which can be computed.

Interestingly, QQP has a much lower difference in proportions and amount of missing trees in the test split. This could be due to the fact that the dataset is the largest in size by a large margin. Potentially, a large amount of data points begins to counteract the granularity of the POS-tree. As with a greater

sample size, you come closer to representing the population grammatical distribution rather than a sample.

In future, I theorise that the attribute-agnostic POS-tree representation could have multiple uses. One possible use could complement the behavioural tests of Checklist [2]. In the introduction, I introduced the idea that it can be difficult to diagnose the reasons for machine learning models' shortcomings. This POS-tree tracking could be used to track which grammatical structures the model has seen during training and then offer alternatives to test on. This project aims to present two grammatical encodings of text and the above results show the POS-tree's uses to be multi-faceted.

4.1.3 Syntactic-Dependencies

Determining Drift Using SDCs

Given two splits, s_1 and s_2 , which contain data points that have all been encoded with the SDC grammatical representation, I perform statistical analysis to determine whether or not s_2 is likely to have been drawn from a separate distribution to s_1 . The nature of the SDC encoding means that each syntactic dependency label is associated with a frequency within a given data point for each textual input. Using the two distributions of SDCs from the two splits, I perform a two-tailed Mann-Whitney U test [76] with a critical value of 0.05 for each syntactic dependency label. Formally, the null and alternative hypotheses are defined in Hypothesis 4.1 and Hypothesis 4.2 respectively.

Hypothesis 4.1. h_0 , the null hypothesis: both splits have been drawn from the same distribution and therefore the frequencies of a given dependency label, l , will be statistically similar in both splits.

Hypothesis 4.2. h_1 : the splits have not been drawn from the same distribution and so the distribution of l in s_2 is different to that of l in s_1 .

As described in Section 4.1, I test tasks in GLUE and SuperGLUE for data drift between the train and test splits. For each label, if the p-value from the Mann-Whitney U test is lower than the critical value of 0.05, I reject the null hypothesis in favour of the alternative hypothesis and assume that the frequencies of that label were not drawn from the same distribution. The larger the number of labels that are deemed out of distribution can therefore be seen as a greater difference in grammatical distribution between the two data splits. In Table 4.3 I show the results as a fraction, $\frac{x}{y}$, where x represents the total number of labels reported as showing drift between the two splits (related to any of the task inputs) and y is the total number of possible dependency labels (45) multiplied by the number of task inputs.

Table 4.3: Dependency labels with drift.

Benchmark	Task	Labels with Drift
SuperGLUE	RTE	0/90
	COPA	5/135
	BoolQ	7/90
	CB	8/90
	WiC	10/90
	MNLI (Matched)	25/90
	WSC	14/45
	MultiRC	68/135
	ReCoRD	51/90
GLUE	MRPC	2/90
	COLA	9/45
	RTE	18/90
	WNLI	22/90
	STS _B	34/90
	QNLI	41/90
	QQP	44/90
	SST-2	31/45

As a control, I also sampled the BookCorpus dataset and performed several random splits on the data. Each of these random splits was then subject to the same analysis as the GLUE and SuperGLUE tasks underwent. Over 1000 different splits, the number of labels with drift between splits has a mean value of 3.56 out of 45 labels and a standard distribution of 2.31. This means that only 2 SuperGLUE tasks and 1 GLUE task contain official splits that contain the same or less grammatical distribution than a single source, but grammatically diverse dataset.

During intermediary stages, I used EvidentlyAI’s dashboard for visual inspection of the dependency drift. See Figure 4.1 for an example.

4.1.4 Conclusion

The share differences in POS-trees were not different enough for any of the subtasks to indicate an out-of-distribution problem. This is likely due to the fact that because the granularity of the POS-tree structure makes it too sensitive to slight changes such as positional changes as described in 3.2.3. Hence the proportions of splits made up by similar structures fluctuate greatly and cause similar results for a variety of tasks.

4 Results

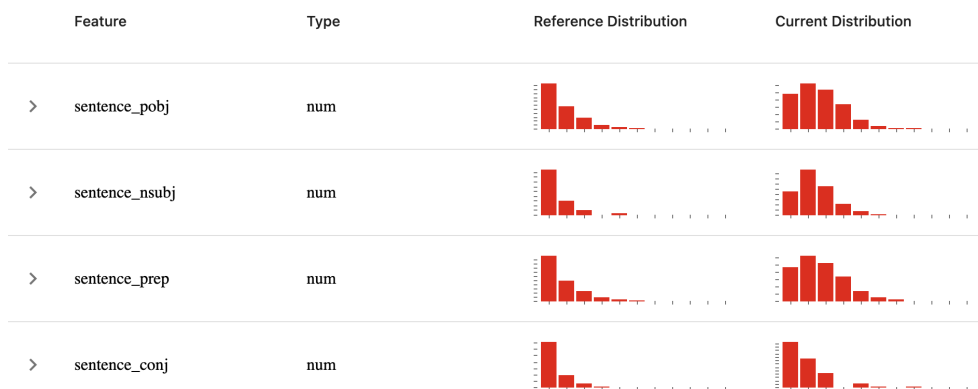


Figure 4.1: Snippet from the EvidentlyAI dashboard visualisation distribution difference in SST-2 train and test splits.

However, the distribution drift detected with the SDC representation is significant, showing that all tasks except one contain some grammatical distribution change between train and test splits. This also highlights how the use cases for each representation differ.

4.2 Investigating Detected Drift

In this section of the results chapter, I explore the possibility that the differences in grammatical distribution found in the previous results section were not the result of random shuffling.

4.2.1 Recreating Splits

Based on the results from the initial drift investigation I analyse a subset of the tasks further. The first step I take is to follow the recreating splits procedure I outlined in Section 3.4.1. I merge the splits and shuffle before recalculating the number of labels with drift repeatedly. This stage is repeated 500 times so that I can utilise the central limit theorem (CLT) [77]. This means that I can create a maximum likelihood estimate (MLE) to fit the data to a normal distribution and the large number of repetitions increases the effectiveness of both CLT and MLE [78]. Given an estimated normal distribution, I am able to estimate the probability that the initial drift was caused by chance. If this probability is below a critical value, 0.01, it suggests that the split was manufactured in a certain way to contain out of distribution data such as by including data from another source in the test split. Formally, the null and alternative hypotheses are defined in Hypothesis 4.3 and Hypothesis 4.4 respectively.

4 Results

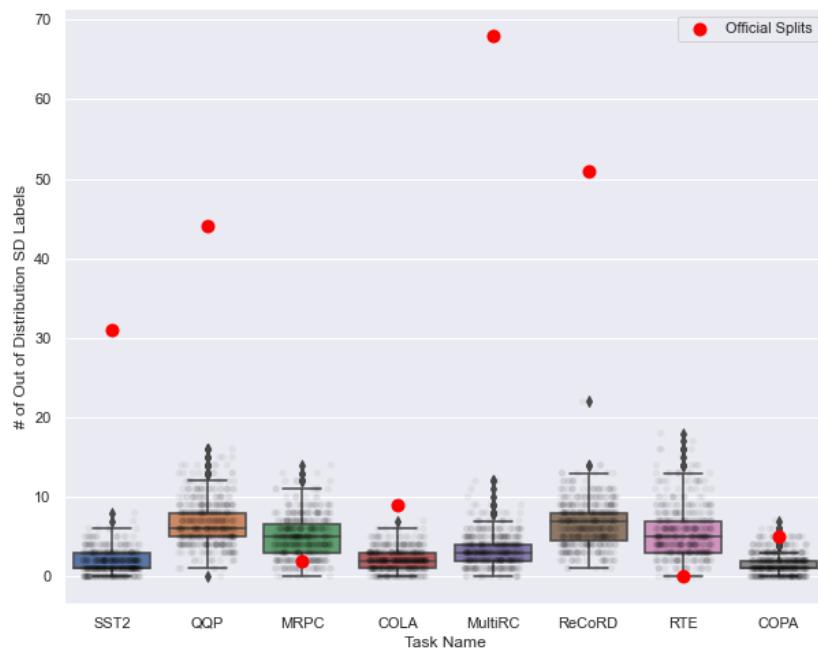


Figure 4.2: Distribution of number of dependency labels calculated as showing grammatical distribution drift between random splits by task (Left: GLUE; Right: SuperGLUE)

Hypothesis 4.3. h_0 , The null hypothesis: the difference in grammatical distribution found between the official test split and train splits is drawn from the same distribution as randomly shuffling the data.

Hypothesis 4.4. h_1 , the alternative hypothesis: the difference in grammatical distribution was the result of non-random splitting and suggests that the splits have been engineered to demonstrate a difference.

Table 4.4: Hypothesis testing for shuffled splits.

Benchmark	Task	P-Value	Accept/Reject H_0
SuperGLUE	RTE	2.65×10^{-02}	Accept
	COPA	1.56×10^{-01}	Accept
	BoolQ	1.14×10^{-01}	Accept
	CB	7.13×10^{-02}	Accept
	WiC	6.18×10^{-08}	Reject
	WSC	4.66×10^{-19}	Reject
	MultiRC	8.76×10^{-76}	Reject
	ReCoRD	1.68×10^{-38}	Reject
GLUE	MRPC	8.12×10^{-02}	Accept
	COLA	4.05×10^{-06}	Reject
	RTE	1.58×10^{-05}	Reject
	WNLI	1.05×10^{-24}	Reject
	STSB	2.43×10^{-21}	Reject
	QNLI	1.98×10^{-39}	Reject
	QQP	5.11×10^{-30}	Reject
	SST-2	4.70×10^{-51}	Reject

4.3 Discussion

4.3.1 Task Specific Findings

WNLI and WSC

Some of the largest train-test split distribution differences appear in the WNLI and WSC subtasks. Both of these tasks are tasks based on the same Winograd Schema Challenge datasets. During the re-splitting stage of my method, I found that it was almost impossible to recreate alternative splits with such great differences in grammatical distribution. This is explained by Kocijan et al. [79] who explain that the subset of Winograd Schema

Challenge datasets used to create these subtasks was split in such a way that the test split includes a previously unreleased set of Winograd Schemas.

QQP

The GLUE splits of the Quora-Question Pairs dataset are the same as the original authors [57]. Interestingly, the authors of GLUE comment that the test split of QQP has a different label distribution to the train split similarly to MRPC. However, the results from my initial benchmark analysis and the split recreation stage show that despite the label distribution difference, the train splits of MRPC do not show a substantial difference in grammatical distribution to the test splits while QQP does. Equally, the official MRPC splits are statistically likely to have been drawn from random shuffling while QQP is not.

CoLA

The findings from analysing CoLA present an unexpected combination of results. My results show that the official CoLA splits do not contain a substantial difference in grammatical distribution. However, the difference in distribution has been shown to be statistically different to the splits of data if they were randomly shuffled. The test split of the Corpus of Linguistic Acceptability task publicly contains out-of-domain data [70] where out-of-domain data is data drawn from different linguistics publications. All of the data, whether reserved as out-of-domain or not, is analysed for linguistic acceptability to find labels for the task during which, several categories of sentences are removed. This cleansing combined with the restricted use of linguistic publications as a source of data could be the reason for such a low difference in grammatical distribution despite the reserved out-of-domain sub-split. The out-of-domain sources were chosen as a result of randomly selecting the sources until the sum of their sizes exceeds 750. The presence of a reserved sub-split equally confirms the findings that the splits were statistically tested to not have been drawn from a random shuffling procedure.

4.3.2 Validity and Relevance

Concerning the validity of the results presented in this section, there are a few notable points. Firstly, the control test on the BookCorpus does provide a baseline to compare the tasks to. However, the mere fact that it produces out of distribution syntactic labels despite being from one data source raises a point of discussion. I hypothesise that the cause of this small change in distribution is because, despite the fact that the data is from one data source, the data is not grammatically controlled. While in tasks such as

question-answering tasks, a ‘question’ attribute is much more likely to remain within a given grammatical distribution rather than a free text attribute. Potentially, the method requires a form of calibration to decide how strictly the grammar should remain within a specific distribution.

The second stage of the analysis, when calculating the probability that the splits were created through random shuffling is the most novel part of the method. There are several methods apart from random shuffling which can be used to create train/test splits. However, alternative methods such as leave-one-out, K-fold cross-validation and stratified sampling should equally not result in grammatical distribution difference unless there is a relationship between grammatical distribution and labels in the data. In which case, this method would bring attention to the fact that whatever method is being used could be restricting the model based on an unintended grammatical relationship. Overall, I believe that the significance of the second stage lies in the ability to raise awareness of possibly unintended grammatical distribution drift between datasets.

Including out of domain data in test splits is a complicated matter. The benchmarks analysed in this project are aimed to evaluate state of the art general language models. Due to the dominance of pre-training models at the time of writing, it is hard to argue that models which have been fine-tuned on the task-specific train splits will have never seen the grammatical constructs used in the test split unless the test split is the result of extreme adversarial dataset generation. Tasks such as the Winograd Schema Challenge in fact, require that the model has prerequisite knowledge to succeed by requiring that the model makes decisions based on a general understanding of the world we live in.

Ultimately, to come closer to knowing whether we impede a model’s performance by introducing grammatical distribution drift between splits would require an investigation of its own with fewer restrictions on computational power.

4.3.3 Conclusion

The GLUE and SuperGLUE benchmarks contain a varying amount of difference in grammatical distributions between official train and test splits. It is also clear that some tasks reserve data from different sources for test splits. Both of these conclusive findings have been confirmed individually in publications either by authors of the tasks, benchmarks or researchers who have analysed the datasets separately.

These results show how I have completed the aims of the project by successfully using the grammatical encodings presented to perform grammatical distribution drift analysis.

4.4 Dependency Analysis

Having analysed the grammatical distribution differences by grouping all of the syntactic dependency labels combined together, I break down the labels to see if certain syntactic dependencies are more commonly the cause of distribution drift than others.

4.4.1 Common OOD Syntactic Dependency Labels

Firstly, I iterate through the features which were recorded as being out of grammatical distribution between the official train and test splits. Then I group the features based on the syntactic dependency label they represent. The frequencies of each label is displayed in Figure 4.3.

It is clear from the bar chart that the most commonly out of domain syntactic dependency label is 'PUNCT'. This could be because punctuation such as commas, and quotation marks can be used to create more complex grammatical structures and hyphens can be used to create compound nouns.

Even though the encoded SDCs could be used to perform a much more in-depth analysis into the relationships between specific syntactic dependency labels in this context, it remains outside the scope of this project and would likely require a project of its own with a greater emphasis on linguistics rather than machine learning.

4.4.2 Conclusion

In this section, I present a small insight into the results found in this project from a task-agnostic perspective, focusing solely on the frequencies and relationships between out-of-grammatical distribution syntactic dependency labels. Importantly, these results are proof of the flexibility of the SDC encoding to complement its use in statistical analysis.

5 Project Conclusion

At the beginning of this project, I set out to use grammatical distribution analysis to present a novel method for detecting distribution drift between textual sources. To demonstrate this, I used the method on benchmarks which are used to evaluate state of the art NLP general language models. I have statistically shown that several of the subtasks contain a substantial difference in grammatical distribution between official train and test splits.

Given computational restrictions, I can confidently say that I achieve the goals I set out at the beginning of the project. I present two different ways to represent the grammatical distributions of NLP datasets and provided a suitable evaluation of the two different representations. To showcase the features of both representations, I include results that add context and provide a greater understanding of the datasets discussed while presenting further use cases for both encodings.

Then I use these representations to evaluate the train/test split distributions of the tasks in GLUE and SuperGLUE, popular NLP benchmarks. Following the initial distribution drift investigation, I also use a novel method to provide insight into the way the datasets have been constructed.

Future Work

There are several directions that future work could take following on from the work in this project. Equipped with more powerful computational resources, one area which would greatly complement this work would be a test involving the fine-tuning of one or more state of the art language models in their base configuration on different split configurations. This could be combined with the analysis of how the splits differ in grammatical distribution to see how grammatical distribution drift affects model performance if at all.

Conversely, a full study of methods use to encode grammatical distributions in datasets could also bring further insight. It would allow investigations such as those conducted during this project to become more commonplace and therefore increase the attention given to linguistic attributes during NLP research which often is discarded in search for pure semantic meaning.

Appendices

A Appendix

Table 5.1: GLUE and SuperGLUE tasks used during the project¹.

Benchmark	Task	Full Title
GLUE	QQP	Quora Question Pairs
	QNLI	Question NLI
	SST	The Stanford Sentiment Treebank
	COLA	The Corpus of Linguistic Acceptability
	STSB	Semantic Textual Similarity Benchmark
	RTE	Recognizing Textual Entailment
	MRPC	Microsoft Research Paraphrase Corpus
	WNLI	Winograd NLI
SuperGLUE	COPA	Choice of Plausible Alternatives
	CB	CommitmentBank
	WiC	Words in Context
	BoolQ	BoolQ
	RTE	Recognizing Textual Entailment
	MultiRC	Multi-Sentence Reading Comprehension
	ReCoRD	Reading Comprehension with Commonsense
	WSC	The Winograd Schema Challenge

Table 5.2: List of syntactic dependency labels used by the SpaCy EN Transformer Model.

Tag Label	Syntactic Dependency
acl	clausal modifier of noun (adjectival clause)
advcl	adverbial clause modifier
advmod	adverbial modifier
agent	agent
amod	adjectival modifier
appos	appositional modifier
attr	attribute
aux	auxiliary

¹See [57] and [58] for details regarding the GLUE and SuperGLUE specific implementations of the tasks.

A Appendix

Tag Label	Syntactic Dependency
auxpass	auxiliary (passive)
case	case marking
cc	coordinating conjunction
ccomp	clausal complement
compound	compound
conj	conjunct
csubj	clausal subject
csubjpass	clausal subject (passive)
dative	dative
dep	unclassified dependent
det	determiner
dobj	direct object
expl	expletive
intj	interjection
mark	marker
meta	meta modifier
neg	negation modifier
nmod	modifier of nominal
npadvmod	noun phrase as adverbial modifier
nsubj	nominal subject
nsubjpass	nominal subject (passive)
nummod	numeric modifier
oprd	object predicate
parataxis	parataxis
pcomp	complement of preposition
pobj	object of preposition
poss	possession modifier
preconj	pre-correlative conjunction
predet	None
prep	prepositional modifier
prt	particle
punct	punctuation
quantmod	modifier of quantifier
relcl	relative clause modifier
xcomp	open clausal complement

Bibliography

- [1] E. Strubell, A. Ganesh and A. McCallum, *Energy and policy considerations for deep learning in nlp*, 2019. arXiv: 1906.02243 [cs.CL].
- [2] M. T. Ribeiro, T. Wu, C. Guestrin and S. Singh, 'Beyond accuracy: Behavioral testing of nlp models with checklist,' in *Association for Computational Linguistics (ACL)*, 2020.
- [3] Z. Yang, A. Zhang and A. Sudjianto, 'Enhancing explainability of neural networks through architecture constraints,' *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2020. DOI: 10.1109/TNNLS.2020.3007259.
- [4] H. Yuan, H. Yu, S. Gui and S. Ji, *Explainability in graph neural networks: A taxonomic survey*, 2021. arXiv: 2012.15445 [cs.LG].
- [5] H. Chefer, S. Gur and L. Wolf, *Transformer interpretability beyond attention visualization*, 2020. arXiv: 2012.09838 [cs.CV].
- [6] G. Brunner, Y. Liu, D. Pascual, O. Richter, M. Ciaramita and R. Wattenhofer, *On identifiability in transformers*, 2020. arXiv: 1908.04211 [cs.CL].
- [7] J. Vig, *A multiscale visualization of attention in the transformer model*, 2019. arXiv: 1906.05714 [cs.HC].
- [8] A. Halevy, P. Norvig and F. Pereira, 'The unreasonable effectiveness of data,' *IEEE Intelligent Systems*, vol. 24, pp. 8–12, 2009. [Online]. Available: http://www.computer.org/portal/cms_docs_intelligent/intelligent/homepage/2009/x2exp.pdf.
- [9] N. Sambasivan, S. Kapania, H. Highfill, D. Akrong, P. K. Paritosh and L. M. Aroyo, '"everyone wants to do the model work, not the data work": Data cascades in high-stakes ai,' 2021.
- [10] E. Breck, M. Zinkevich, N. Polyzotis, S. Whang and S. Roy, 'Data validation for machine learning,' in *Proceedings of SysML*, 2019. [Online]. Available: <https://mlsys.org/Conferences/2019/doc/2019/167.pdf>.
- [11] C. G. Northcutt, A. Athalye and J. Mueller, *Pervasive label errors in test sets destabilize machine learning benchmarks*, 2021. arXiv: 2103.14749 [stat.ML].

Bibliography

- [12] K. Hao, *Error-riddled data sets are warping our sense of how good ai really is*, M. T. Review, Ed., Apr. 2021. [Online]. Available: <https://www-technologyreview-com.cdn.ampproject.org/c/s/www.technologyreview.com/2021/04/01/1021619/ai-data-errors-warp-machine-learning-progress/amp/>.
- [13] K. Burns, L. A. Hendricks, K. Saenko, T. Darrell and A. Rohrbach, *Women also snowboard: Overcoming bias in captioning models*, 2019. arXiv: 1803.09797 [cs.CV].
- [14] N. Garg, L. Schiebinger, D. Jurafsky and J. Zou, 'Word embeddings quantify 100 years of gender and ethnic stereotypes,' *Proceedings of the National Academy of Sciences*, vol. 115, no. 16, E3635–E3644, Apr. 2018, ISSN: 1091-6490. DOI: 10.1073/pnas.1720347115. [Online]. Available: <http://dx.doi.org/10.1073/pnas.1720347115>.
- [15] J. Zhao, T. Wang, M. Yatskar, V. Ordonez and K.-W. Chang, *Men also like shopping: Reducing gender bias amplification using corpus-level constraints*, 2017. arXiv: 1707.09457 [cs.AI].
- [16] J. Buolamwini and T. Gebru, 'Gender shades: Intersectional accuracy disparities in commercial gender classification,' in *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, S. A. Friedler and C. Wilson, Eds., ser. Proceedings of Machine Learning Research, vol. 81, New York, NY, USA: PMLR, Feb. 2018, pp. 77–91. [Online]. Available: <http://proceedings.mlr.press/v81/buolamwini18a.html>.
- [17] T. DeVries, I. Misra, C. Wang and L. van der Maaten, *Does object recognition work for everyone?* 2019. arXiv: 1906.02659 [cs.CV].
- [18] S. Shankar, Y. Halpern, E. Breck, J. Atwood, J. Wilson and D. Sculley, 'No classification without representation: Assessing geodiversity issues in open data sets for the developing world,' in *NIPS 2017 workshop: Machine Learning for the Developing World*, 2017.
- [19] B. Hutchinson, V. Prabhakaran, E. Denton, K. Webster, Y. Zhong and S. Denuyl, *Social biases in nlp models as barriers for persons with disabilities*, 2020. arXiv: 2005.00813 [cs.CL].
- [20] O. Sharir, B. Peleg and Y. Shoham, *The cost of training nlp models: A concise overview*, 2020. arXiv: 2004.08900 [cs.CL].
- [21] V. Gwiasda, N. Taluc and S. J. Popkin, 'Data collection in dangerous neighborhoods: Lessons from a survey of public housing residents in chicago,' *Evaluation Review*, vol. 21, no. 1, pp. 77–93, 1997. DOI: 10.1177/0193841X9702100105. eprint: <https://doi.org/10.1177/0193841X9702100105>. [Online]. Available: <https://doi.org/10.1177/0193841X9702100105>.
- [22] R. Dale, 'Gpt-3: What's it good for?' *Natural Language Engineering*, vol. 27, no. 1, pp. 113–118, 2021. DOI: 10.1017/S1351324920000601.

Bibliography

- [23] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, *Language models are few-shot learners*, 2020. arXiv: 2005.14165 [cs.CL].
- [24] Y. Zheng, G. Chen and M. Huang, *Out-of-domain detection for natural language understanding in dialog systems*, 2020. arXiv: 1909.03862 [cs.CL].
- [25] T. Mikolov, K. Chen, G. Corrado and J. Dean, *Efficient estimation of word representations in vector space*, 2013. arXiv: 1301.3781 [cs.CL].
- [26] J. Pennington, R. Socher and C. D. Manning, 'Glove: Global vectors for word representation,' in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: <http://www.aclweb.org/anthology/D14-1162>.
- [27] N. Reimers and I. Gurevych, *Sentence-bert: Sentence embeddings using siamese bert-networks*, 2019. arXiv: 1908.10084 [cs.CL].
- [28] G. D. F. Jr., 'The viterbi algorithm: A personal history,' *CoRR*, vol. abs/cs/0504020, 2005. arXiv: cs/0504020. [Online]. Available: <http://arxiv.org/abs/cs/0504020>.
- [29] T. Brants, 'Tnt - A statistical part-of-speech tagger,' *CoRR*, vol. cs.CL/0003055, 2000. [Online]. Available: <https://arxiv.org/abs/cs/0003055>.
- [30] L. Màrquez, L. Padró and H. Rodríguez, 'A machine learning approach to pos tagging,' *Machine Learning*, vol. 39, no. 1, pp. 59–91, Apr. 2000, ISSN: 1573-0565. DOI: 10.1023/A:1007673816718. [Online]. Available: <https://doi.org/10.1023/A:1007673816718>.
- [31] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa, *Natural language processing (almost) from scratch*, 2011. arXiv: 1103.0398 [cs.LG].
- [32] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký and S. Khudanpur, 'Recurrent neural network based language model,' vol. 2, Jan. 2010, pp. 1045–1048.
- [33] T. Mikolov, S. Kombrink, L. Burget, J. Černocký and S. Khudanpur, 'Extensions of recurrent neural network language model,' in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 5528–5531. DOI: 10.1109/ICASSP.2011.5947611.

Bibliography

- [34] I. Sutskever, J. Martens and G. Hinton, 'Generating text with recurrent neural networks,' in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11, Bellevue, Washington, USA: Omnipress, 2011, pp. 1017–1024, ISBN: 9781450306195.
- [35] S. Liu, N. Yang, M. Li and M. Zhou, 'A recursive recurrent neural network for statistical machine translation,' in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 1491–1500. DOI: 10.3115/v1/P14-1140. [Online]. Available: <https://www.aclweb.org/anthology/P14-1140>.
- [36] M. Auli, M. Galley, C. Quirk and G. Zweig, 'Joint language and translation modeling with recurrent neural networks,' in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1044–1054. [Online]. Available: <https://www.aclweb.org/anthology/D13-1106>.
- [37] I. Sutskever, O. Vinyals and Q. V. Le, *Sequence to sequence learning with neural networks*, 2014. arXiv: 1409.3215 [cs.CL].
- [38] T. Robinson, M. Hochberg and S. Renals, 'The use of recurrent neural networks in continuous speech recognition,' Jan. 1995, ISBN: 978-1-4612-8590-8. DOI: 10.1007/978-1-4613-1367-0_10.
- [39] A. Graves, A.-r. Mohamed and G. Hinton, *Speech recognition with deep recurrent neural networks*, 2013. arXiv: 1303.5778 [cs.NE].
- [40] A. Graves and N. Jaitly, 'Towards end-to-end speech recognition with recurrent neural networks,' in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14, Beijing, China: JMLR.org, 2014, II–1764–II–1772.
- [41] H. Sak, A. Senior and F. Beaufays, *Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition*, 2014. arXiv: 1402.1128 [cs.NE].
- [42] A. M. Dai and Q. V. Le, *Semi-supervised sequence learning*, 2015. arXiv: 1511.01432 [cs.LG].
- [43] B. McCann, J. Bradbury, C. Xiong and R. Socher, *Learned in translation: Contextualized word vectors*, 2018. arXiv: 1708.00107 [cs.CL].
- [44] M. E. Peters, M. Neumann, L. Zettlemoyer and W.-t. Yih, *Dissecting contextual word embeddings: Architecture and representation*, 2018. arXiv: 1808.08949 [cs.CL].
- [45] T. Young, D. Hazarika, S. Poria and E. Cambria, *Recent trends in deep learning based natural language processing*, 2018. arXiv: 1708.02709 [cs.CL].

Bibliography

- [46] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, *Attention is all you need*, 2017. arXiv: 1706.03762 [cs.CL].
- [47] J. Howard and S. Ruder, *Universal language model fine-tuning for text classification*, 2018. arXiv: 1801.06146 [cs.CL].
- [48] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 [cs.CL].
- [49] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, *Roberta: A robustly optimized bert pretraining approach*, 2019. arXiv: 1907.11692 [cs.CL].
- [50] W. Zeng, X. Ren, T. Su, H. Wang, Y. Liao, Z. Wang, X. Jiang, Z. Yang, K. Wang, X. Zhang, C. Li, Z. Gong, Y. Yao, X. Huang, J. Wang, J. Yu, Q. Guo, Y. Yu, Y. Zhang, J. Wang, H. Tao, D. Yan, Z. Yi, F. Peng, F. Jiang, H. Zhang, L. Deng, Y. Zhang, Z. Lin, C. Zhang, S. Zhang, M. Guo, S. Gu, G. Fan, Y. Wang, X. Jin, Q. Liu and Y. Tian, *Pangu-alpha: Large-scale autoregressive pretrained chinese language models with auto-parallel computation*, 2021. arXiv: 2104.12369 [cs.CL].
- [51] P. He, X. Liu, J. Gao and W. Chen, *Deberta: Decoding-enhanced bert with disentangled attention*, 2021. arXiv: 2006.03654 [cs.CL].
- [52] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, 'Language models are unsupervised multitask learners,' 2019.
- [53] G. Mesnil, Y. Dauphin, X. Glorot, S. Rifai, Y. Bengio, I. Goodfellow, E. Lavoie, X. Muller, G. Desjardins, D. Warde-Farley, P. Vincent, A. Courville, J. Bergstra, I. Editor, G. Guyon, V. Dror, G. Lemaire, D. Taylor and D. Silver, 'Unsupervised and transfer learning challenge: A deep learning approach,' vol. 7, pp. 1–15, Jan. 2011.
- [54] D. Bahdanau, K. Cho and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, 2016. arXiv: 1409.0473 [cs.CL].
- [55] Y. Kim, C. Denton, L. Hoang and A. M. Rush, *Structured attention networks*, 2017. arXiv: 1702.00887 [cs.CL].
- [56] A. P. Parikh, O. Täckström, D. Das and J. Uszkoreit, *A decomposable attention model for natural language inference*, 2016. arXiv: 1606.01933 [cs.CL].
- [57] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy and S. R. Bowman, *Glue: A multi-task benchmark and analysis platform for natural language understanding*, 2019. arXiv: 1804.07461 [cs.CL].
- [58] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy and S. R. Bowman, *Superglue: A stickier benchmark for general-purpose language understanding systems*, 2020. arXiv: 1905.00537 [cs.CL].

Bibliography

- [59] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng and C. Potts, 'Recursive deep models for semantic compositionality over a sentiment treebank,' *EMNLP*, vol. 1631, pp. 1631–1642, Jan. 2013.
- [60] B. Pang and L. Lee, 'Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales,' in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, Ann Arbor, Michigan: Association for Computational Linguistics, Jun. 2005, pp. 115–124. DOI: 10.3115/1219840.1219855. [Online]. Available: <https://www.aclweb.org/anthology/P05-1015>.
- [61] D. Klein and C. D. Manning, 'Accurate unlexicalized parsing,' in *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan: Association for Computational Linguistics, Jul. 2003, pp. 423–430. DOI: 10.3115/1075096.1075150. [Online]. Available: <https://www.aclweb.org/anthology/P03-1054>.
- [62] S. Ackerman, E. Farchi, O. Raz, M. Zalmanovici and P. Dube, *Detection of data drift and outliers affecting machine learning model performance over time*, 2021. arXiv: 2012.09258 [stat.AP].
- [63] A. Van Looveren, G. Vacanti, J. Klaise and A. Coca, *Alibi-Detect: Algorithms for outlier and adversarial instance detection, concept drift and metrics*. Version 0.6.2, 2019. [Online]. Available: <https://github.com/SeldonIO/alibi-detect>.
- [64] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba and S. Fidler, *Aligning books and movies: Towards story-like visual explanations by watching movies and reading books*, 2015. arXiv: 1506.06724 [cs.CV].
- [65] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P. J. Liu, *Exploring the limits of transfer learning with a unified text-to-text transformer*, 2020. arXiv: 1910.10683 [cs.LG].
- [66] T. H. Trinh and Q. V. Le, *A simple method for commonsense reasoning*, 2019. arXiv: 1806.02847 [cs.AI].
- [67] A. Williams, N. Nangia and S. R. Bowman, *A broad-coverage challenge corpus for sentence understanding through inference*, 2018. arXiv: 1704.05426 [cs.CL].
- [68] R. May, H. Maier and G. Dandy, 'Data splitting for artificial neural networks using som-based stratified sampling,' *Neural Networks*, vol. 23, no. 2, pp. 283–294, 2010, ISSN: 0893-6080. DOI: <https://doi.org/10.1016/j.neunet.2009.11.009>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0893608009002949>.
- [69] O. Rozen, V. Shwartz, R. Aharoni and I. Dagan, *Diversify your datasets: Analyzing generalization via controlled variance in adversarial datasets*, 2019. arXiv: 1910.09302 [cs.CL].

Bibliography

- [70] A. Warstadt, A. Singh and S. R. Bowman, *Neural network acceptability judgments*, 2019. arXiv: 1805.12471 [cs.CL].
- [71] B. Dolan and C. Brockett, 'Automatically constructing a corpus of sentential paraphrases,' in *Third International Workshop on Paraphrasing (IWP2005)*, Asia Federation of Natural Language Processing, Jan. 2005. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/automatically-constructing-a-corpus-of-sentential-paraphrases/>.
- [72] M.-C. de Marneffe, M. Simons and J. Tonhauser, 'The commitment-bank: Investigating projection in naturally occurring discourse,' *Proceedings of Sinn und Bedeutung*, vol. 23, no. 2, pp. 107–124, Jul. 2019. DOI: 10.18148/sub/2019.v23i2.601. [Online]. Available: <https://ojs.uni-konstanz.de/sub/index.php/sub/article/view/601>.
- [73] O. Honovich, L. Choshen, R. Aharoni, E. Neeman, I. Szpektor and O. Abend, *Q2: Evaluating factual consistency in knowledge-grounded dialogues via question generation and question answering*, 2021. arXiv: 2104.08202 [cs.CL].
- [74] S. Reddy, Y. Yu, A. Pappu, A. Sivaraman, R. Rezapour and R. Jones, *Detecting extraneous content in podcasts*, 2021. arXiv: 2103.02585 [cs.CL].
- [75] Y. Scherrer and N. Ljubešić, 'Social media variety geolocation with geoBERT,' in *Proceedings of the Eighth Workshop on NLP for Similar Languages, Varieties and Dialects*, Kiyv, Ukraine: Association for Computational Linguistics, Apr. 2021, pp. 135–140. [Online]. Available: <https://www.aclweb.org/anthology/2021.vardial-1.16>.
- [76] G. Upton and I. Cook, *A Dictionary of Statistics*, en, 2nd ed. Oxford University Press, Jan. 2008, ISBN: 9780199541454. DOI: 10.1093/acref/9780199541454.001.0001. [Online]. Available: <http://www.oxfordreference.com/view/10.1093/acref/9780199541454.001.0001/acref-9780199541454> (visited on 18/05/2021).
- [77] 'Central limit theorem,' in *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, pp. 66–68, ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1_50. [Online]. Available: https://doi.org/10.1007/978-0-387-32833-1_50.
- [78] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, p. 132, <http://www.deeplearningbook.org>.
- [79] V. Kocijan, T. Lukasiewicz, E. Davis, G. Marcus and L. Morgenstern, *A review of winograd schema challenge datasets and approaches*, 2020. arXiv: 2004.13831 [cs.CL].