MSc. DSAI

Assessing the capabilities of self-explainable neural networks on gender text classification

Internship report

Presented and defended on August 26th, 2022

by

Mariana CHAVES

Company supervisors: Lucile Sassatelli and Frédéric Precioso

I3S and Inria

Contents

Abstract

Artificial Neural Networks have demonstrated astonishing potential to absorb natural language skills, offering a new and rich path to study human speech. However, harvesting the knowledge incorporate in those models has proven to be a major challenge since their sheer scale prevents any direct interpretation of their working. The recently proposed architecture *ProtoryNet* promises to provide itself access to its functioning. This is an immensely valuable trait for both research and applications, and in this work we investigate those promises. Namely, we study ProtoryNet in the challenging context of gender classification of movie characters. To that end, we construct a sizeable dataset of labelled character dialogues based on the Cornell corpus and use it to benchmark ProtoryNet and many other reference machine learning models.

Our results indicate that, while the accuracy of ProtoryNet is on par with that of well-established deep models, it does not provide superior explanations when compared to classic intrinsically interpretable models.

Introduction

"Life imitates Art far more than Art imitates Life"
– Oscar Wilde, *The Decay of Lying*

Gender roles are social constructs and, as such, are influenced by the media. They are malleable sets of ideas, shaped in large part by observation of pre-established models. Cinema is perhaps the most important exposition of those models. Not only for its immense reach but also for its multimedia nature. Movies are bound to quickly communicate ideas to broad audiences and, to do so, they commonly resource to distilled representations of cultural models, in all of their aspects. In particular, they absorb and present back to us a condensed version of how genders should sound, look, and behave, shaping our collective imagination and influencing our attitudes towards these ideas. Unfortunately, those distillations are prone to rely on archetypes and simplifications [1] and sometimes screenwriters resource solely on stereotypes [2].

The progress of artificial intelligence and machine learning has offered us a way to study the patterns that govern gender representation in media large scales. Several authors have used those tools to perform gender studies on movie scripts [3, 1, 4, 2] and other media [5, 6, 7, 8]. They bring to light how stereotypical gender roles and biases are conveyed by those means.

Many works in this field defined the patterns they want to measure beforehand, resourcing to feature engineering, semantic fields, and other similar latent variables to characterise dialogues of male and female characters. Instead, we propose to learn those motifs.

We focused on gender classification based on text data via *self-explainable neural networks*. These models output not only a solution to the task at hand but also an explanation of their decision [9], making them a promising tool for understanding patterns of gender in speech.

More precisely, this work studies the *ProtoryNet* model [10]. It belongs to the group of self-explaining neural networks for sequential data, hence, being suitable for text classification. The network learns *prototypes*: data points or parts of data points that are exemplar cases for each category. It makes predictions by comparing the inputs to those prototypes. For example, if the input is more similar to the prototypes related to the category "female", it is expected for the model to predict "female."

My[1] main objective consisted in assessing the capabilities of ProtoryNet, both in terms of its accuracy and the clarity of its explanations. The machine learning task consisted in classifying "male" and "female" film characters based on their speech. To this end, I assembled a sizeable dataset for the task based on existing films corpora, the main source being the Cornell corpus [11]. To properly evaluate the performance of ProtoryNet I compared it with other text classification models. I trained a variety of machine learning models, from intrinsically interpretable models (logistic regression and complement naïve bayes) to deep models regarded as black-boxes (fully connected networks, LSTMs, Bidirectional LSTMs and BERT models).

This study is part of the TRACTIVE (acronym for TowaRds A Computational mulTImodal analysis of film discursiVe aEsthetics) initiative. TRACTIVE is a project supported by Université Côte d'Azur, Inria, I3S, Sorbonne Université, Université Toulouse 3, Université Sorbonne Paris Nord, and CNRS, and its main objective is "to characterise and quantify gender representation and women objectification in films and visual media, by designing an AI-driven multimodal (visual and textual) discourse analysis."[2] The project explores many aspects of gender representation in media. One of those focuses on assessing the text classification capabilities of prototype-based deep neural networks. The current study lies on that framework.

Chapter 1 provides a glance at gender studies that used machine learning, as well as the terminology related to interpretability, self-explainable machine learning, and prototype-based models. Chapter 2 describes the data collection process, based on the Cornell corpus. Also, I use word embedding models and the Word-Embedding Association Test (WEAT) [12] to provide a preliminary analysis of the gender biases present in the dataset. Subsequently, I present performance results for different text classification models in Chapter 3. Finally, in Chapter 4 I explain ProtoryNet's characteristics and evaluate its accuracy and the quality of its explanations.

[1]Following the indication of the reviewing committee, first person singular (*e.g.*, "I,", "my.") is used to indicate the author's independent analysis or contribution, while first person plural (*e.g.*, "we") refers to activities where other members of the group took part in the implementation.

[2]`https://www.i3s.unice.fr/TRACTIVE/`

1

Related work

1.1 Data science in gender studies

Data science enabled the analysis of gender patterns in large corpora. Motifs that pass undetected by humans can be capture by machine learning models. In [6], the authors use a corpus of emails and support vector machines to determine the gender of the author of each document. The work [7] uses transcriptions of conversations and machine learning approaches (specifically logistic regression and BERT models) to explore distributional differences in language use across gender. Convolutional and recurrent neural networks were used to automatically annotate gender of mentions in the text in [8]. They apply these tools to movie summaries, movie reviews, news articles, and fiction novels. The authors of [5] use support vector machines, Bayesian logistic regression and AdaBoost decision trees to identify the gender of the author of news and emails. In [1] natural language processing techniques are applied on a corpus of films to extract gender patterns. The work [4] creates a corpus based on screenplays and collect statistics related to gender using feature engineering. In [2] logistic regression and naïve bayes classifiers are trained to distinguishing the gender of movie characters. While in [3], they implement a computational approach to automate the task of finding whether a movie passes or fails the Bechdel test. A movie passes the test if there at least two named female characters that talk to each other about something besides a male character. They show that movies that fail the test are highly likely to have less meaningful rolls for women.

Those previous studies unveil differences in grammatical structures used by each gender, such as women's tendency to use emotionally intensive and affective adverbial and adjectival expressions, such as "really," "so," "very," "quite," "adorable," and "charming."

Whereas men's conversational patterns aim to retain status and attention. For instance, they use more first-person singular pronouns such as "I" and more directive sentences [1, 5, 2]. In [4] the authors identify that female character utterances tend to be more positive and that male characters seem to have higher percentage of words related to achievement. Also, polite words such as greetings and "please" are favoured by women, while cursing is a male-favoured practice [1, 2, 4, 7]. They show that women's lexical fields in movies are more related to shopping, cleaning, personal care, and family, whereas men's are associated with money, sports, work, and male friendship [1]. The work in [7] identifies tendencies of women to use minimal particles such as "my God," "mhm," "blah," "huh," and "hm", while men tend to use "ain't," "innit," "eh," and "uh." Also, among the top terms used by male speakers in spoken conversation they identified the nouns "mate," "game," "quid," and "football", while for female speakers the identified nouns were "baby," "weekend," "hair," "birthday," and "cake."

Other works such as [12, 13] show that neural networks absorb human biases, including gender related ones.

1.2 Interpretability

1.2.1 What is interpretability and why is it important

Interpretability is defined as "the degree to which a human can understand the cause of a decision"[14], or consistently predict a system's result [15]. Unfortunately, there is no consolidated mathematical definition to measure such degree. In terms of models, one is considered more interpretable than another if its decisions are easier for a human to comprehend [15]. Therefore, the explanations of a model are constrained to be human-understandable.

The importance of interpretability lies in the fact that if the model can explain decisions we can check its fairness (ensuring that predictions are unbiased and do not implicitly or explicitly discriminate against under-represented groups), privacy (ensuring that sensitive information in the data is protected), reliability (ensuring that small changes in the input do not lead to large changes in the prediction), causality (check that only causal relationships are picked up) and trust, as it is easier for humans to trust a system that explains its decisions [15]. In [16], the authors makes this last point evident with the hypothetical example of a machine learning model used for medical diagnosis or terrorism detection, whose predictions cannot be acted upon on blind faith, since its consequences could be of high impact.

There is the need to evaluate the model as a whole before deployment. Usually, models are evaluated using accuracy metrics. However, a model can provide the right answer for the wrong reasons [17]. The European Union's General Data Protection Regulation (GDPR) requires companies to be able to provide an explanation about decisions made by artificial intelligences [9]. Therefore interpretability of machine learning models is becoming mandatory.

1.2.2 Taxonomy of interpretability methods

The different characteristics of interpretability methods allow us to categorise them.

Intrinsic or *post-hoc* interpretability. Intrinsic interpretability refers to models that are interpretable due to their structure. The classic examples include simple models such as decision trees, generalised linear models (GLMs), and generalised additive models. For instance, in linear regression, where the predictions $\hat{y} \in \mathbb{R}^n$ are determined by

$$\hat{y} = \theta_0 + \theta_1 x_1 + \ldots + \theta_p x_p = \theta X,$$

where each $x_i \in \mathbb{R}^n$ is an input variable, $\theta_i \in \mathbb{R}$, the estimated coefficient related to x_i, and θ_0, the intercept. Given the linear relationships between the features and the prediction, we know that an increase of one unit in the i-th feature, x_i, increases $\hat{y}$ by θ_i, if we keep all the other features constant. This interpretation assumes that x_k is a numerical variable, but similar interpretations can be done for categorical ones. The key insight of this example is that it is its structure that makes the model intrinsically interpretable.

Note that, intrinsic interpretability is not limited to simple models. For instance, SENN [18], ProtoPNet [19], and ProSeNet [20] offer deep network approaches that are intrinsic interpretable. The authors of ProtoryNet [10] (our model of focus) imply that this model also belongs to this category, nevertheless I challenge this idea in Section 4.3.3.

Post hoc interpretability, on the other hand, refers to the application of interpretation methods after model training [15]. These methods are mostly used to explain black-box models. Due to the limited access to the inner workings of the model, *post hoc* techniques usually use gradients or reverse propagation, (ε-Layerwise Relevance Propagation (E-LRP) [21], Integrated Gradients [22], and Grad-CAM [23]), or they create surrogate models that capture the input-output behaviour (LIME [16], kernel Shapley values (SHAP) [24]). They approximate the behaviour of the black-box and/or show trends in how predictions are related to the features.

Model-specific or model-agnostic. Model-specific methods are limited to a specific type of model. For instance, the interpretation of regression coefficients cannot be directly applied to a neural network model. Examples for deep models are saliency maps [25] and DeepLIFT [26] which can only be applied to neural networks. ProtoryNet is model-specific. On the contrary, model-agnostic tools can be used on any machine learning model and are applied after the model has been trained. These methods do not have access to model internals such as weights or structural information [15].

Local or global interpretability. Interpretability methods can explain an individual prediction (local interpretability) or the entire model behaviour (global interpretability). Global model interpretability explains how parts of the model affect the predictions. For example, linear regression provides global interpretability, as its interpretations apply for all instances. Local interpretability aims to answer why the model made a specific prediction. ProtoryNet has global interpretability.

1.2.3 Self-explaining machine learning

Self-explaining machine learning models are intrinsic interpretability models which use machine learning. They yield two outputs: the decision of the model and an explanation of that decision [9]. ProtoryNet is said to belong to this class of models.

The importance of self-explainability.

Studies on interpretability of complex machine learning models has mostly focused on estimating *a posteriori* explanations for previously trained models [18], *i.e.* the use of *post-hoc* methods. Although *post-hoc* techniques can throw some light over the input-output relations of a black-box model or mimic the calculations made by it, since they do not completely unveil the inner workings of the model, some authors [27] claim that black-box approaches should not be used for high-stake decision making, such as healthcare and criminal justice. Instead, models with intrinsic interpretability should be applied.

The work [27] opts for self-explaining machine learning techniques over using black-box models in conjunction with *post-hoc* techniques. It states that

- it is a myth that there is necessarily a trade-off between accuracy and interpretability. Intrinsic interpretable models can often provide as good accuracy results as their non-interpretable counterparts. For instance, the experimental results in [20] and [19] support this statement by comparing their self-explaining models against state-of-the-art deep learning models.

- *Post-hoc* methods provide explanations that are not faithful to what the original model computes. If they were, the explanation would be equivalent to the original model.

- *Post-hoc* explanations can provide ambiguous or nor stable results. For instance, to intend to explain models used for image classification, saliency maps create heat maps that can be overlapped with an image. The heatmap highlights the parts of the image that are used by the classifier to take its decision. Nevertheless, the saliency maps for multiple classes could be essentially the same or very similar. The explanation for why the image is classified as Siberian husky can be the same as the explanation for being classified as a transverse flute, as shown in [27].

Prototype-based models.

Among the self-explaining models, there are the prototype-based models. ProtoryNet is part of this group. These approaches learn prototypes to provide its predictions and explanations. By prototypes, I refer to data points or parts of data points that are exemplar cases for a category. To make a prediction, these models compare a data point against the prototypes. For instance, consider the MNIST [28] image classification scenario, where the machine learning task is to classify images of handwritten digits. The prototypes for the category "number 8" can be some images, or sections of images, from the training set that well represent this category. For text classification, prototypes can be sentences, phrases, or words extracted from the documents in the training set.

I detail how ProtoryNet uses prototypes for its predictions in Section 4.1. In the following paragraphs, I discuss other prototype-based models that predated it.

A rich class of interpretable models is formalised in [18]. They design self-explaining models in stages, progressively generalizing linear classifiers to more complex models. As part of this family, they introduce Self-Explaining Neural Networks (SENN). Where, instead of using the raw inputs for the explanations, more interpretable features are used. These more general features are called *interpretable basis concepts*. For instance, when using images as data points, instead of making interpretations for each pixel it is better to interpret in terms of strokes. The model is defined as

$$f(x) = \sum_{i=1}^{K} \theta(x)_i h(x)_i, \tag{1.1}$$

where $x_1, \ldots, x_n \in \mathbb{R}$ are raw input variables, θ is a deep neural network, and $h(x) : \mathcal{X} \to \mathcal{Z} \subset \mathbb{R}^k$. $\mathcal{X}$ is the space of raw inputs, and $\mathcal{Z}$, an a space of interpretable basis concepts. The key is that, among others, h can be based on prototypes.

Prototypical part network (ProtoPNet) [19] is another deep network that works with prototypes in image classification. It dissects the image to find prototypical parts for each class and makes its prediction based on a weighted combination of the similarity scores between parts of the image and the learned prototypes. The similarity scores can be understood as to how strongly a prototypical part is present in some patch of the input image. The network consists of a convolutional neural network, followed by a prototype layer and a fully connected layer. Each prototype can be understood as the latent representation of some prototypical part of a class.

The work [29] proposes a similar self-explaining neural network. Its architecture contains an autoencoder and a special prototype layer, where each unit of that layer stores a weight vector that resembles an encoded training input. Given the use of an autoencoder, a corresponding decoder is necessary for visualizing the prototypes. This work represents a particular case of the family of models formalised before [18].

ProSeNet [20] is a self-explaining model designed to work with sequence data. This work is the most similar to our model of interest, ProtoryNet. ProSeNet generates a determined number of prototypes for each class. The model consists of three parts:

- A recurrent sequence encoder network that maps each input sequence to an embedding vector. The encoder could be any backbone sequence learning model, for instance, an Long Short-Term Memory network (LSTM) [30]. The output at the last step of the LSTM, the last hidden state, is used as the embedding vector.
- The prototype layer that contains k prototypes and measures the similarity between the embedded sequences and the prototypes.
- A fully connected layer with a final softmax layer for output probabilities in multi-class classification tasks.

The model aims to satisfy three criteria for constructing the prototypes: simplicity, diversity, and sparsity. *Simplicity* aims to create simple and short prototypes. *Diversity* focuses on avoiding redundant prototypes and prefers those that are sufficiently distinct from each other. *Sparsity* establishes that for each input only a few prototypes should be "activated." In Section 4.1, you can observe that ProtoryNet also intends to enforce these desiderata in the definition of its loss function and its the architecture.

2

The Cornell Corpus

My initial experiments on ProtoryNet were performed on a corpus of movie dialogues of similar size to the corpus of positive and negative reviews used in the original work. These first attempts resulted in poor accuracies, which I hypothesized to be an effect of increasing the task complexity without providing significantly more data for the training. Intuitively, while its generally easy for a human to identify if a review is positive or negative, it is uncommon for the same text to make clear the gender of the author. To test this hypothesis, I opted for a much larger dataset: the Cornell corpus [11]. This is a collection of 304713 utterances extracted from film scripts along with details about the speaker and the movie to which each dialogue belongs. It provides data for 9035 characters from 617 movies. Particularly, it includes gender information for 3015 of the characters. It also indicates which dialogues belong to each conversation and which characters participated on it.

2.1 Data preparation

Data labelling

Since the gender of many of the characters is not labelled in the Cornell corpus, I used information from the TMDB 5000 Movie Dataset[3] to automatically fill 588 of the missing labels. The automatic procedure attempts to match the name of the character and movie between the two datasets, hence being limited by inconsistencies between them. Ultimately, I manually annotated gender data for 3806 characters using information from

[3]Available on the Kaggle platform at https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata?select=tmdb_5000_credits.

the TMDB website[4]. Combined, the three sources provided labelled data 7409 characters, corresponding to 94% of the dialogues available in the Cornell corpus.

Data preprocessing

I based the data preprocessing on the work [2], which uses the Cornell corpus to train non-deep models to classify characters as male or female based on their dialogues and identify linguistic and structural features of dialogue that differentiate speech by gender. The classification was done at character level, that is, each sample corresponds to all dialogues of a character. I use the same approach. In their preprocessing steps, they made an effort to avoid confounding factors that could yield spurious significant results. Guided by their scope I:

- Used dialogues from characters who had at least 3 conversations with other characters, 10 utterances, and 100 words spoken in total. This was intended to consider characters that have a non-trivial amount of speech in the film.
- Randomly selected equal numbers of male and female characters from each movie, to obtain a balanced dataset.
- After removing punctuation, considered unigrams that appeared at least 5 times, obtaining a vocabulary of 11233 unique words. I did not apply stemming, lemmatisation, or stop-word removal since we are interested not only in the lexical meaning but also in the forms used by each gender. For instance, previous studies show that women use more adverbial language compared to men [1, 5, 4, 2].

These filters preserved 2404 characters of the 9035 in the original Cornell dataset, belonging to 567 movies. These characters represent 51% of the original dialogues.

Cross-validation approach

To control for the style of screenwriters, the authors of [2] use leave-one-*group*-out when training their gender classification models, where the group is determined by the movie. This is a special case of cross-validation, similar to leave-one-out. When applying leave-one-out, each time, one sample is used as test set while all other samples are used for training. Thus, there are as many folds as data samples. In contrast, when using leave-one-group-out (in this case, leave-one-*movie*-out), each time, the samples in one of the

[4]https://www.themoviedb.org/

groups are used as the test set and the remaining groups are used for training. That is, each cross-validation split takes the characters of one movie and designate them as the test set. Hence, the number of folds is equal to the number of movies.

Nevertheless, this approach is not viable when the training of the model is very time consuming. For instance, for ProtoryNet models, each epoch takes around 20 minutes (using a GRID RTX6000-24C GPU). Therefore, training a model for 20 epochs using leave-one-movie-out would take approximately 157 days. For this reason, I randomly assigned each movie to one of 5 groups and used those for the leave-one-group-out. In other words, each cross-validation split takes the samples from one of those 5 groups of movies and uses them as the test set. This approach still provides some control for the style of screenwriters and topics associated with each movie while keeping the number of folds small (just 5). I will refer to this approach as *leave-some-movies-out*.

Leave-some-movies-out is used to asses the accuracy of all the text classification models presented in Chapters 3 and 4. There are cases where, besides a train and test sets, a validation one is also required. For instance, consider the training of a fully connected network where we do not want to save the model the last epoch, but rather at the moment when the validation loss was the smallest. The final model is determined by both the training and validation sets. In this case, for each cross-validation split, one group of movies is reserved as testing set. From the remaining samples, 80% are assign to training and 20%, to validation.

2.2 Gender biases in the corpus

When training neural networks on text data, the format of the input comprises a challenging on its own since usual text encodings ignore the semantic aspect of language. For example, consider the words “cat” and “car”. Even though they differ by a single character, the word “cat” is semantically much further from “car” than from “kitten”, which does not share a single letter with “cat”.

Ideally, before feeding the input to the model the words would be mapped into a space in which distances between points reflect the semantic similarity between the respective words. In the context of the previous example, it is desirable for the points associated with “cat” and “car” to be further apart than the points associated with “cat” and “kitten”. Such space is called a semantic space and a mapping of words into a vector space is called a word embedding.

Fortunately, neural networks can learn such mapping as long as enough data is available for the training. However, an embedder network itself is passive to interpretation since the learned representation absorbs a lot of information from the dataset, as we will see. Those embedder models represent more of a preprocessing step, and, thus, present relatively simpler architectures, training algorithms, and tasks compared to other NLP models powered by NLP. Therefore, they tend to preserve more direct relationships with the statistics of the dataset. Those features can be leveraged to extract valuable statistical measures of the training data from the model. Among the many techniques that can be used for this task, I chose to use Word2vec [31, 32] for this project. This choice was based on its relatively low computational requirements and on the fact that it does not require specially large datasets. Moreover, the principles behind this method are reasonably intuitive, so I provide an introduction to those in the next paragraphs.

First of all, a key strength of the Word2vec is that it does not rely on any manual labelling. This means it requires no human experts at all. In technical terms, Word2vec is an instance of unsupervised learning (more precisely, of self-supervised learning). This immensely reduces the cost of training data, making massive datasets available.

The strategy is to leverage a poor word embedding and huge amount of training data to obtain a good encoding. The simplest way to produce an initial embedding is to use the dictionary of all the words[5] present in the dataset as encoded space. This can be achieved by associating each dictionary entry with a dimension. Then one can translate a given piece of text into a vector by counting how many times each word occurs in the piece and indicating this amount in the respective coordinate. For instance, suppose the word “friend” corresponds to position number 555 of the vector, “my” to position 444, and “little” to position 123. Then the text “my little friend” would be converted into a vector with 1 in positions 123, 444 and 555, and 0s in all the other positions.

A few notes are pertinent at this point:

- This is not the final word embedding. This simple encoding is known as *Bag-of-Words representation*;

- In practice those vectors have huge dimensionality since the dictionary tends to be large. As a reference, the Cornell corpus produces a dictionary with roughly 11,233 thousand entries;

[5]The term “tokens” would be more general than “words”, but the latter was preferred for the sake of didactics.

- Short samples of text result into vectors consisting mostly of zeros as most text use a small fraction of the words in the dictionary;
- This representation completely ignores grammar and even the order in which words appear.

Now we need a way for the model to "train itself" to extract semantic information from text. The problem is that, at this point, no tool connected to semantics is available yet. To work around this limitation, one can employ word placement as a proxy for word meaning: if two words are used in similar contexts, they should be semantically close.

In practice, the network is trained in the following routine:

- Iterate through the dataset, word by word;
- Encode the current word and give it as input to the model;
- Try to predict the neighbourhood (its corresponding vector) of the original word, for example, the 5 previous and 5 next words (as a Bag-of-Words representation).

This training routine is known as *skip-gram*. I preferred it to the alternatives as, according to the original work [31], it performs better for infrequent words.

Since the answer for this query is known (the words nearby the target), it is possible to use it to improve the network's guess. This does not provide the desired semantic embedding directly, but it can be found it in the intermediate layers of the neural network. The architecture is such that the information flows through a relatively narrow point. To give an idea, the architecture employed takes in the data through more than 10 thousand input neurons and forces that through only 100 neurons at some point. The activations at this bottleneck is our final semantic embedding.

Surprisingly, this works very well. Indeed, this the semantic space obtained not only has the desired property of keeping embeddings of semantically close words geometrically close to each other, but it also behaves almost as a semantic vector space. This means it is possible to perform operations with words and get a meaningful result. For example, for such semantic space obtained, we have

$$\text{vector}(\text{“england”}) - \text{vector}(\text{“london”}) + \text{vector}(\text{“tokyo”}) \approx \text{vector}(\text{“japan”})$$

and

$$\text{vector}(\text{“man”}) - \text{vector}(\text{“power”}) \approx \text{vector}(\text{“woman”}).$$

As the last example indicates, the encoding of words itself carries human biases. Since the task of gender classification is expected to be connected with such biases, it would be desirable to have an idea of how much of those the encoder preserves. Quantifying such subjective traits is, of course, a major challenge and many approaches to it exist in the literature. For human subjects, perhaps the most famous one is the *Implicit Association Tests (IAT)* [33] which attempts to reveal and measure the strength of the connection between concepts through individuals' behaviour. The IAT relies on the hypothesis that humans tend to perform better in association tasks when those align with preconceived biases. For example, subjects should be quicker to tag the word "insect" with the label "disgusting" than with the label "pleasant". The IAT proposes to quantify this phenomenon as a proxy for implicit biases.

It is possible to obtain similar measures for our embedder with an analogous statistical test, the *Word-Embedding Association Test (WEAT)* [12]. When translating the IAT paradigm to word embedders, the WEAT can exploit the fact that for these models the association between words is quite evident since the semantic space is built to encode such relations in its geometry. The method, thus, reduces to computing statistics similar to those employed by the IAT using a suitable concept of distance in the semantic space. More precisely, for our case, the test considers four groups of words, two with terms that convey a target concept (e.g. "flowers" and "insects") and two representing attributes (e.g. "beauty" and "disgust"). Statistics are collected on the distance[6] of pairs of words, one from a target group and one for an attribute group. Finally, the effect size is expressed in terms of Cohen's d, with conventional small, medium, and high values being 0.2, 0.4, and 0.8, respectively.

The work [12] detects strong biases in encoders pre-trained in many popular datasets. So, to provide a clearer picture of the Cornell corpus, I train a Word2vec model from scratch on it. Table 2.1 compiles our results.

The pair "musical instruments"/"weapons" and "pleasant"/"unpleasant" is used as a control, as associations between the pairs are well documented [33]. While the test still measures a strong bias towards musical instruments being pleasant and weapons being unpleasant, the effect size is considerably weaker than that found for humans ($d = 0.82$ vs. $d = 1.66$, respectively). I remark that many of the words in the list of musical instruments from [33] did not occur in our dataset frequently enough to be included in the embedder. Following [12], I compensated for that by removing the same amount of words from the

[6]In our context, the cosine similarity.

Target words	Attribute words	Effect size (d)	p-value
Male vs. Female terms	Science vs. Arts	0.48	0.17
Male vs. Female terms	Career vs. Family	0.42	0.20
Male vs. Female names	Family vs. Career	0.48	0.18
Male vs. Female names	Arts vs. Science	0.96	0.03
Instruments vs. Weapons	Pleasant vs. Unpleasant	0.82	0.02

Table 2.1: WEAT results for the Word2vec embedding utilised for this project.

other three groups associated to that query uniformly at random. I provide the exact lists of words used in Table 1.

Our probes for gender biases indicate the existence of moderate to strong tendencies in the encoding of words. However, I did not obtain enough statistical significance for most of them with the notable exception of the target groups “male names”/“female names” with attributes “arts”/“science”. There, perhaps surprisingly, the test detects a strong bias towards the association of male names to concepts related to arts relative to the association of female names to words related to science.

3

Text classification models

Properly assessing the capabilities of ProtoryNet requires comparison with other text classification models. In this chapter I present performance results for different intrinsically interpretable models (logistic regression and complement naïve bayes) and deep models regarded as black-boxes (fully connected networks, LSTMs, Bidirectional LSTMs and BERT models).

Recall that the task at hand consists in classifying characters as “male” or “female” based on their utterances. Each sample corresponds to all dialogues of a character, preprocessed as described in Section 2.1. *Leave-some-movies-out* is applied as cross-validation method.

3.1 Interpretable non-deep models

In [2] the authors used the Cornell corpus to train logistic regression and naïve bayes models to distinguish between male and female characters based on their dialogues. Those models are of interest due to their intrinsic interpretability, which I discuss later in this section. I replicated part of the results described in [2] and added some variations, explained below.

Logistic regression and naïve bayes models are not suitable for raw text inputs, so each text sample must be encoded in a vector representation. I applied two of such encodings: Bag of words (BoW), which I described in Section 2.2, and *term frequency–inverse document frequency (TF-IDF)* [34], a classical numerical statistic related to the importance of a word to a text in a corpus. I experimented those using only unigrams, and using unigrams, bigrams and trigrams. In the latter setup, each position of the vector represen-

Model	Average test accuracy ± stddev. (%)		
	General	Male	Female
Unigrams			
TF-IDF + LR	69.49 ± 2.36	71.07 ± 2.55	67.91 ± 2.96
BoW + LR	67.88 ± 1.18	64.86 ± 1.77	70.89 ± 0.95
TF-IDF + CNB	69.70 ± 0.87	70.49 ± 1.81	68.90 ± 1.53
BoW + CNB	68.90 ± 0.59	86.39 ± 1.37	51.41 ± 1.82
1, 2, and 3-grams			
TF-IDF + LR	**72.09 ± 1.46**	74.71 ± 3.69	69.47 ± 1.85
BoW + LR	69.66 ± 1.18	69.60 ± 3.81	69.73 ± 1.99
TF-IDF + CNB	70.43 ± 0.90	67.32 ± 1.45	73.54 ± 1.23
BoW + CNB	70.89 ± 1.25	75.14 ± 2.07	66.64 ± 1.63

Table 3.1: Logistic regression (LR) and complement naïve bayes (CNB) models' accuracy results using *leave-some-movies-out* cross-validation.

tation is related either to a word or to a group of words. For example, position number 16 could be related to "my," position number 29 to "my little," and position number 64 to "my little friend." Following the strategy in [2], I considered only n-grams that appeared 5 or more times, obtaining a vocabulary of 11233 words for unigrams, and 65309 n-grams for unigrams, bigrams, and trigrams.

For naïve bayes models, I applied the *Complement Naïve Bayes (CNB)* algorithm [35], which is an adaptation Multinomial Naïve Bayes (MNB). The work [35] empirically supports that the parameter estimates for CNB are more stable than those for MNB and that CNB regularly outperforms it on text classification tasks.

I use the scikit-learn [36] implementation of logistic regression and CNB, tuning the hyperparameters through grid search. The best combinations found are reported in the Appendix (Tables 2 and 3).

Table 3.1 shows that "simple" non-deep models can achieve test accuracies between 67.88% and 72.09%. Models using unigrams, bigrams and trigrams perform better than their counterparts that use only unigrams. This is expected since the first ones have more information at their disposal. Although the best performing model uses logistic regression, its test accuracy confidence interval overlaps with two of the CNB models. Hence, there is not a clear winner between logistic regression and CNB. Also, note that one of the CNB

Top				Bottom			
n-gram$_i$	e^{θ_i}	n-gram$_i$	e^{θ_i}	n-gram$_i$	e^{θ_i}	n-gram$_i$	e^{θ_i}
he	1.67	⋮	⋮	*the*	0.60	⋮	⋮
hes	1.50	*of me*	1.32	*got*	0.63	*my wife*	0.76
my husband	1.43	*oh*	1.31	*on*	0.69	*she*	0.76
so	1.42	*cant*	1.31	*here*	0.69	*running*	0.76
with him	1.38	*god*	1.31	*hey*	0.70	*shes*	0.76
my god	1.37	*him*	1.30	*right*	0.70	*understand*	0.77
are you	1.36	*what is it*	1.30	*of*	0.72	*hey you*	0.77
are	1.36	*should*	1.27	*her*	0.73	*there is*	0.77
just	1.35	*awful*	1.27	*there*	0.73	*up*	0.77
nice	1.35	*cant believe*	1.27	*yeah*	0.74	*could*	0.77
youre just	1.33	*dont*	1.28	*man*	0.75	*problem*	0.78
husband	1.33	*silly*	1.28	*gotta*	0.76	*its the*	0.78
oh my	1.32	*please*	1.26	*her to*	0.76	*shit*	0.78
⋮	⋮	*we were*	1.25	⋮	⋮	*business*	0.79

Table 3.2: n-grams related to the top and bottom coefficients of the best performing logistic regression model, along with their exponentiated coefficients. n-grams related to top coefficients are more linked to the category "female," while those related in the bottom are more related to the category "male."

models (BoW + CNB applied to unigrams) is clearly skewed towards the class "male."

Given that the best performing model uses logistic regression, let me detail the type of interpretations that the model offers. Recall that such model is determined by

$$\ln\left(\frac{\Pr(y=1)}{1-\Pr(y=1)}\right) = \ln\left(\frac{\Pr(y=1)}{\Pr(y=0)}\right) = \theta_0 + \theta_1 x_1 + \ldots + \theta_p x_p,$$

where n is the number of data points, $y \in \{0,1\}^n$, each $x_i \in \mathbb{R}^n$ is an input variable, $\theta_i \in \mathbb{R}$ is the estimated coefficient related to x_i, and θ_0 is the intercept. The term inside the logarithm is the odds of belonging to class 1. Thus,

$$\begin{aligned}\text{odds} &= \frac{\Pr(y=1)}{\Pr(y=0)} \\ &= \exp(\theta_0 + \theta_1 x_1 + \ldots + \theta_i x_i + \ldots + \theta_p x_p) \\ &= e^{\theta_0} \cdot e^{\theta_1 x_1} \cdots e^{\theta_p x_p}.\end{aligned}$$

Now, increasing the i-th input variable by one unit, we have

$$\begin{aligned} e^{\theta_0} \cdot e^{\theta_1 x_1} \cdots e^{\theta_i (x_i+1)} \cdots e^{\theta_p x_p} &= e^{\theta_0} \cdot e^{\theta_1 x_1} \cdots e^{\theta_i x_i} \cdot e^{\theta_i} \cdots e^{\theta_p x_p} \\ &= e^{\theta_i} \cdot \text{odds}. \end{aligned}$$

That is, all other variables constant kept constant, an increase of one unit to the i-th feature changes the odds of belonging to class 1 by a factor of e^{θ_i}. Note that if $\theta_i = 0$, then $e^{\theta_i} = 1$, and, hence, the estimated odds are unchanged.

Consider now the model with the best test accuracy (logistic regression on the TF-IDF representation of unigrams, bigrams and trigrams). Dialogues by male and female speakers were coded with 0s and 1s, respectively. The word "he" is linked to the highest coefficient, whose exponential is 1.67. This means that for every increase in one unit of the TF-IDF score of the word "he" the odds that a text belongs to a female speaker augments by 67% (this is equivalent to say that it augments by a factor of 1.67). On the other hand, the word "the" is linked to the smallest coefficient, whose exponential is 0.6. Hence, for every increase in one unit of its TF-IDF score the odds that a text belongs to a female speaker diminishes by 40%. More broadly, words with higher coefficients are more related to the category "female," while those with smaller coefficients are more linked to the category "male."

Table 3.2 presents the most associated n-grams to each category, which match some of the results of [2, 5, 1, 4, 3]. There is a predominance of references to women in men's speech and vice-versa. For instance, "her," "her to," "she," and "my wife" appear among the words related to the male speakers, while "he," "my husband," "with him," and "him" relate to female speakers. Women use more adverbial and adjectival expressions, such as "so," "nice," "awful," and "silly." In contrast, men are more inclined towards expressions related to nouns, which could explain the presence of the article "the." Also polite words (*e.g.*, "please") as more favoured by women, while cursing is a male-favoured practice. The words "business" and "running" show the relation of male speech with lexical fields such as money, sports and work. Women's speech includes more interjectional phrases such as "my god" and "oh."

3.2 Non-interpretable deep models

In this section I explore the capabilities of deep neural network models of different architectures to distinguish male and female characters based on their speech. In contrast to the previous section, I do not explore the possible explanations associated to the models. Although *post-hoc* interpretability methods (*e.g.* LIME [16]) can be used to approximate the inner workings of these models, such methods lie out of the scope of this study. The models were implemented in Keras [37].

3.2.1 Fully connected networks, LSTMs and Bidirectional LSTMs

I experimented with different architectures to find good models for our classification task. All of them start with an embedding layer that transforms each word in a text sequences into a vector of a semantic latent space. That is, if the text sequence contains m words and the latent space is of dimension ℓ, the embedder generates a tensor $\mathsf{E} \in \mathbb{R}^{m \times \ell}$. I explored the following architectures:

1. **Embedding + GlobalMaxPooling + Dense layers + softmax/sigmoid**: the embedding layer is followed by a *Global Max Pooling* function, which computes the maximum for each column of E, yielding a vector of size ℓ. Subsequently, two dense layers are applied. Finally, either the sigmoid or the softmax function is used.

2. **Embedding + LSTM + Dense layer(s) + softmax/sigmoid**: the output of the embedder, the tensor E, serves as input to a *Long Short-Term Memory (LSTM)* layer [30]. The last hidden state of the LSTM (which is a vector of size 64) goes through one or two dense layers. Finally, either the sigmoid or the softmax function is used.

3. **Embedding + Bidirectional LSTM + Dense layer(s) + softmax/sigmoid**: This architecture has the same structure as the previous one with the exception that instead of using a simple LSTM layer, it uses a Bidirectional [38] one.

4. **Embedding + LSTM + GlobalMaxPooling + softmax/sigmoid**: This architecture is a variation of the second one where all hidden states of the LSTM are the input of a global max pooling function.

I experimented with latent spaces of size 64 or 128 and multiple sizes for the dense layers. As for the activation function, I employed either the *Rectified Linear Unit (ReLU)* [39] or the *Gaussian Error Linear Unit (GELU)* [40], which are two of the most popular ones. The loss function was either binary cross-entropy or categorical cross-entropy. If the former was used, the output layer uses sigmoid activation and the target variable is given as a one dimensional binary array. If the later is used, the output layer uses softmax activation

Model	Average test accuracy ± stddev. (%)		
	General	Male	Female
E_{64} + BLSTM + D_{64} + D_{32} + softmax	**67.36 ± 1.31**	67.58 ± 4.96	67.13 ± 2.46
E_{128} + LSTM + D_{64} + D_{32} + softmax	65.29 ± 1.88	60.18 ± 11.75	70.41 ± 8.56
E_{64} + BLSTM + D_{64} + D_{32} + sigmoid	63.98 ± 4.55	73.09 ± 8.24	54.88 ± 15.53

Table 3.3: Best three deep models' accuracy results using *leave-some-movies-out* cross-validation. "E_ℓ" denotes an embedding layer and "D_d" a dense layer, where the subscript indicates the dimensionality of the output. The three architectures employ *GELU* activation function for the dense layers. The LSTM and Bidirectional LSTM layers (BLSTM) have output dimensionality of 64.

and the target variable is a one-hot encoded version of the one dimensional binary array. Those variations amount to 84 combinations. The models were trained using classic train, validation, and test splitting, with the final model corresponding to the epoch when its validation loss was the lowest. Their accuracy results are available in the Appendix in Table 4.

From the 84 models, the best three were selected to be trained again, this time using *leave-some-movies-out* cross-validation. The results are presented in Table 3.3. All of the models perform worse than the best interpretable models from Section 3.1. Also, one of the models is unbalanced, in the sense that it is much better at classifying men's speech than women's speech. Figure 3.1 shows the progress of the training for those three models. Note, that after a few epochs (see Figure 3.1), the validation loss reaches its lowest point. After that point, the validation accuracy stagnates and the training accuracy shows signs of overfitting.

3.2.2 BERT models

Bidirectional Encoder Representations from Transformers (BERT) models [41] are a popular baseline for large NLP models. Their name comes from its use of Transformer encoder architecture to process each token of input text in the full context of all tokens before and after. BERT models are usually pre-trained on a large corpus of text, then fine-tuned for specific tasks, which is the approach I followed. Specifically, I employed the BERT model pre-trained on English Wikipedia and BooksCorpus [42] datasets. Among the pre-trained models available at *Tesorflow Hub*, I deemed this to be the one with a pre-training closest

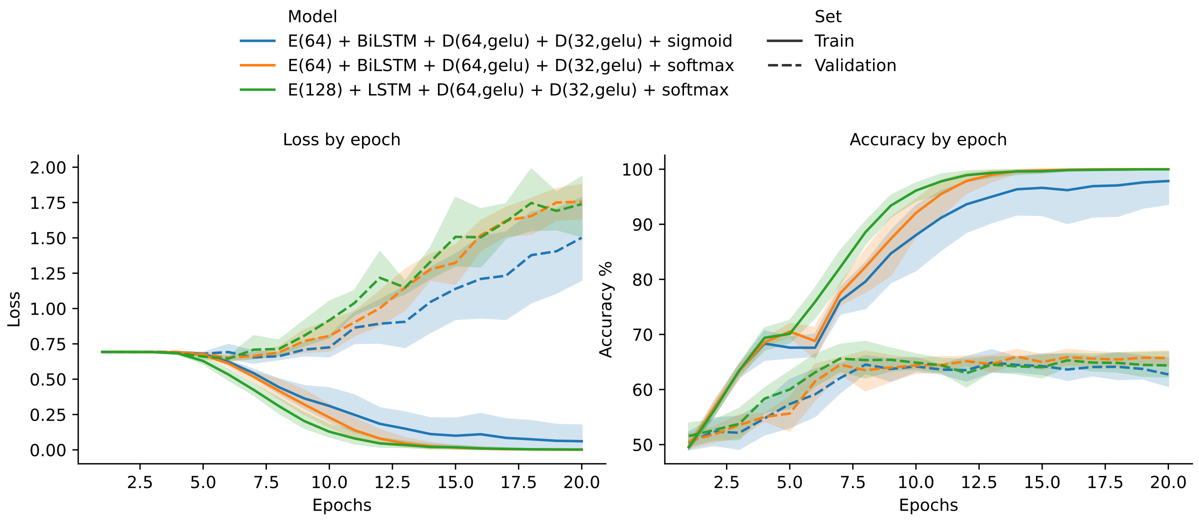

Figure 3.1: Accuracy percent and loss on the train and validation sets by epoch for the best three deep models using leave-one-group-out cross-validation. The center line shows the average and the shaded area represents a 95% confidence interval.

Model	Average test accuracy ± stddev. (%)		
	General	Male	Female
BERT (modified text)	59.02 ± 6.71	89.93 ± 6.64	28.10 ± 20.03
BERT (original text)	**65.17 ± 1.66**	81.15 ± 5.43	49.18 ± 7.80

Table 3.4: BERT models' accuracy results using *leave-some-movies-out* cross-validation.

to our eventual fine-tuning on the preprocessed Cornell corpus dataset.

Two BERT models were fine-tuned. One used as input the original text and the other the altered version of the dialogues (as described in Section 2.1) where punctuation and words that appear less than 5 times were removed. The weights of the final model are obtained by rewinding them to the values that achieved the lowest validation loss. Figure 3.2 depicts the progress of the training for both models, indicating signs of overfitting. The second model shows considerably better test accuracy results (see Table 3.4), which was expected given that, compared to the altered text, the original one is more similar to the natural language on which the BERT model was pre-trained. Moreover, using the modified text, the loss and accuracy present more variability (see Figure 3.2). Nevertheless, both models present poor accuracy for the category "female", which makes them untrustworthy.

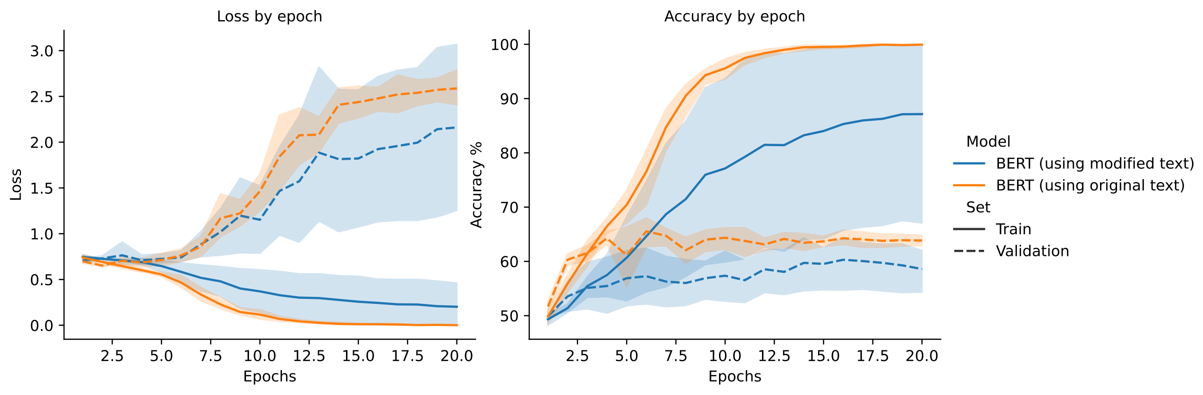

Figure 3.2: Accuracy percent and loss on the train and validation sets by epoch for BERT models using *leave-one-group-out* cross-validation. The center line shows the average and the shaded area represents a 95% confidence interval.

4

ProtoryNet

ProtoryNet [10] is a self-explaining model designed to work with sequential data. As its predecessors mentioned in Section 1.2.3, it compares the inputs to generated prototypes to make predictions. However, it aims to avoid some issues with other prototype-based text classification models.

First, it intends to be more robust when working with long text sequences. Previous approaches, such as ProSeNet [20], define prototypes at the document level. This can make it difficult to match the input sequence to prototypes when those are long and complex documents. For instance, consider a task of classifying reviews as positive or negative and assume that the input document is the four sentences we see in Figure 4.1. Part of the document expresses positive comments, while the rest does not. Such ambivalence can complicate the process of finding adequate prototypes for this input. ProtoryNet attempts to overcome this by splitting the document in sentences and mapping each of them to a single prototype, which is called the *active prototype* of the sentence. For example, the first sentence, “Food was delicious in this place”, is mapped to the prototype, “Great food”.

Second, ProtoryNet aims to simplify the explanations by reducing the number of prototypes needed. Methods such as ProSeNet need hundreds of prototypes to achieve reliable performance, and the explanation of each prediction depends on all prototypes related to a class. In comparison, ProtoryNet only needs as many prototypes as sentences in the input sequence to provide the explanation. As a reference, the work [10] reports that ProtoryNet required only 20 prototypes to match the accuracy of ProSeNet with 200 prototypes.

Moreover, the prototypes for ProSeNet [20] and ProtoPNet [19] belong to a specific class. In contrast, ProtoryNet's prototypes are linked to the classes through a score. For instance, in a binary classification task where a positive review is identified with 1 and a negative one with 0, if a prototype's score is 0.98, then it is mostly associated to positive reviews, while a score of 0.04 indicates its association with negative reviews.

In the rest of this chapter I detail the architecture, loss function and other characteristics of ProtoryNet, I discuss about the impact of different prototype initialisation techniques, I present the performance of ProtoryNet in our text classification task and challenge the interpretability of its results.

4.1 Architecture

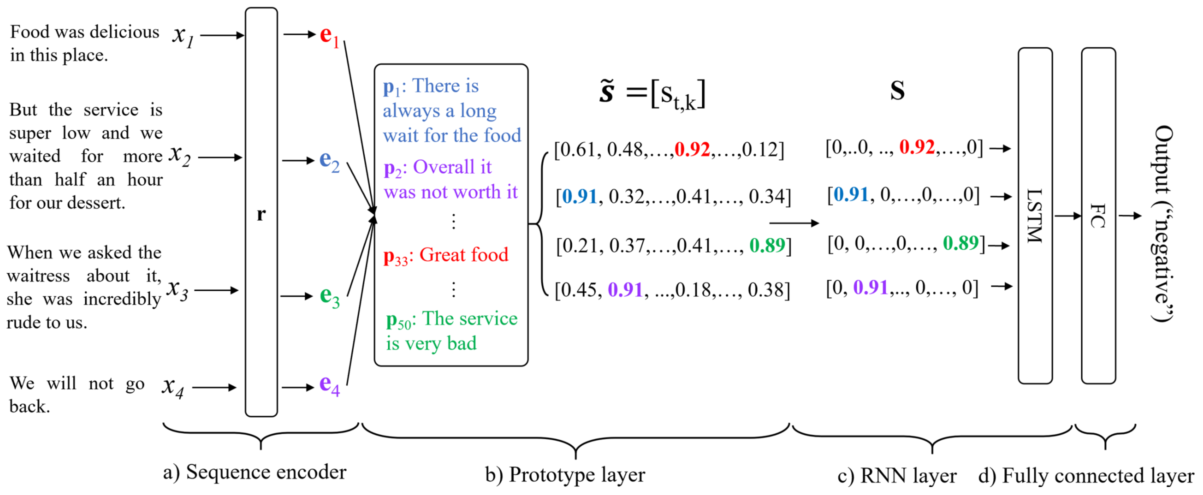

Figure 4.1: Architecture of ProtoryNet from [10].

Consider a data set $\mathcal{D} = \{(X^{(i)}, y^{(i)}) : i = 1, \dots, N\}$. Each $X^{(i)}$ is a text sequence composed of sentences $x^t \in \mathbb{R}^V$, where V is the size of the vocabulary. In other words, $X^{(i)} = (x^t)_{t=1}^T$, with T being the number of sentences in the sequence. Each $y^{(i)} \in \mathbb{R}^C$ is a one-hot encoded vector denoting the class labels and C is the number of classes. The model consists of three parts:

1. **Sentence encoder** r**.** For a given sentence $x^t \in X^{(i)}$, the sentence encoder $r : \mathbb{R}^V \to \mathbb{R}^J$ maps each sentence into a vector e_t of length J. In [10] the authors use the pre-trained Google Universal Encoder [43] as encoder r. I use the same approach and train the encoder along with the rest of the model.

2. **Prototype layer.** This layer contains a set of K trainable prototypes

$$\mathcal{P} = p_k \in \mathbb{R}^J : k = 1, \ldots, K.$$

Note that, the prototypes are representations in the latent space. To make them interpretable, every few epochs, they assign p_k with the closest sequence embedding in the training set. To initialise the prototypes, all sentences in the training set are encoded and then grouped using k-medoids clustering [44]. The resulting medoids are used as the initial prototypes. The prototype layer measures the similarity $s_{t,k}$ between e_t and each prototype p_k. The similarity is defined by

$$s_{t,k} = \exp\left(-\frac{d(e_t, p_k)}{\psi^2}\right),$$

where $\psi \in \mathbb{R}$ was set to $\psi^2 = 10$, and d is the Euclidean distance. The output of the layer is the similarity matrix $\tilde{S} = [s_{t,k}] \in \mathbb{R}^{T \times K}$. Note that $\tilde{S} = [\tilde{s_1}, \ldots, \tilde{s_T}]^\top$, where $\tilde{s_t} \in \mathbb{R}^{\mathbb{K}}$ is the row vector whose elements indicate how similar x_t is to each prototype.

3. **RNN and fully connected layers.** From now on, for each sentence, the architecture only propagates information about the prototype with the maximum similarity to the sentence, the *active prototype*. An immediate way of achieving this would be, for each row of $\tilde{S}$, to zero all but its the maximum entry. However, such sparsification is not differentiable, and therefore not adequate for training. Hence, the authors approximate it by using that $S \approx \Gamma \odot \tilde{S}$, where $\odot$ denotes the Hadamard product and

$$\Gamma = [\text{softmax}(\gamma \cdot \tilde{s_1}), \ldots, \text{softmax}(\gamma \cdot \tilde{s_T})],$$

with some constant $\gamma \geq 1e^6$. Finally, an LSTM model is applied to each row of S, followed by a few fully connected layers.

4.2 Loss function

The loss function encompasses three terms. First, to ensure *accuracy* they minimise the mean squared loss between predicted and ground truth values:

$$\mathcal{L}_{acc}(\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} \|y^{(i)} - \hat{y}^{(i)}\|^2.$$

Second, to guarantee diversity, *i.e.*, to avoid redundant prototypes, a *diversity loss* term is included:

$$\mathcal{L}_{div}(\mathcal{D}) = \sigma(\eta(\delta - d_{min})),$$

where $d_{min} = \min_{k_1,k_2} d(p_{k_1}, p_{k_2})$, d is the Euclidean distance, σ is the sigmoid function, η is a smoothing constant set to 1, and δ is a positive real that enforces a minimum distance among the prototypes.

Finally, a *prototypicality loss* is included:

$$\mathcal{L}_{proto} = \frac{1}{M} \sum_{X^{(i)} \in \mathcal{D}} \sum_{x_t \in X^{(i)}} \min_k d(r(x_t), p_k).$$

It encourages prototypes to be close to the sentences they represent:

The final loss function is $\mathcal{L} = \mathcal{L}_{acc} + \alpha\mathcal{L}_{div} + \beta\mathcal{L}_{proto}$. The authors empirically set $\alpha = 0.1$ and $\beta = 1e^{-4}$, claiming that different combinations of α and β lead to similar performance as long as they are set to small positive values.

4.3 Experiments

4.3.1 Prototype initialisations

As stated before, prototypes for ProtoryNet are initialised by the means of a k-medoids clustering. However, using this approach for our dataset produced redundant prototypes. For instance, in a model of 10 prototypes, k-medoids selected the following initial prototypes: “Oh, God,” “I couldn’t believe it he just left!,” “Oh God,” “Aw, come on,” “You come up with that yourself?,” “Oh, come on,” “Oh my God!,” “Oh dear,” “Oh, dear,” and “Oh no.” In initial experiments, the lack of diversity persisted after training for many epochs. This might indicate that when starting with repeated prototypes it is hard for the model to acquire prototype heterogeneity.

As an alternative, I decided to initialise the prototypes uniformly at random, without replacement. This approach brings the risk of a “bad random selection” of prototypes that will prevent the model from learning properly. As with other algorithms that can be affected by their initialisation, we can run the model several times, see how the accuracy measures behave, and keep the model whose initialisation ended in better results. It is likely that more robust approaches can be designed. Those and a more robust study on the impact of different initialisation are left as future work.

4.3.2 Performance on gender classification

I explored ProtoryNet models with 10, 30, and 50 prototypes. They were trained on the original dialogues, *i.e.*, preserving punctuation and words that appear less than 5 times, to resemble more the data on which Google Universal Encoder was pre-trained. *Leave-some-movies-out* cross-validation was used, saving each model at the moment its validation accuracy was the highest.

Its was expected for the accuracy to increase with the number of prototypes. However, Table 4.1 shows that the models with more prototypes perform the worst. One could hypothesize that after some optimal number of prototypes, any extra would only add noise in the model, harming performance. For our case, the results suggest that this optimal lies around 10 prototypes, which is surprisingly small for a task as complex as identifying gender patterns of speech.

Model	Average test accuracy ± stddev. (%)		
	General	Male	Female
Best Logistic Regression	**72.09 ± 1.46**	**74.71 ± 3.69**	**69.47 ± 1.85**
Best deep (non-interpretable)	67.36 ± 1.31	67.58 ± 4.96	67.13 ± 2.46
ProtoryNet			
10 prototypes	66.18 ± 1.54	74.06 ± 5.30	58.31 ± 4.36
30 prototypes	62.51 ± 1.94	58.30 ± 12.19	66.71 ± 9.33
50 prototypes	55.02 ± 3.09	52.45 ± 12.97	57.59 ± 12.76

Table 4.1: Accuracy results of ProtoryNet models using *leave-some-movies-out* cross-validation alongside those for the best performing logistic regression and deep models.

Figure 4.2 illustrates the validation accuracy during the training progress for the three models. In particular, it indicates that the model with 50 prototypes stagnate slightly above 50% of accuracy. Moreover, the model of 10 prototypes improves only for the first epochs, followed by a gentle decrease in accuracy.

The best ProtoryNet model (the one with 10 prototypes) is outperformed by both the best logistic regression and the best deep non-interpretable models (see Table 4.1). Therefore, the logistic regression model is preferred by all metrics: it is intrinsically interpretable, it offers the best test accuracy results, and its training time is the shortest, specially compared to ProtoryNet. These findings hint that, despite common belief, non-interpretable models do not necessarily outperform intrinsically interpretable ones. Works

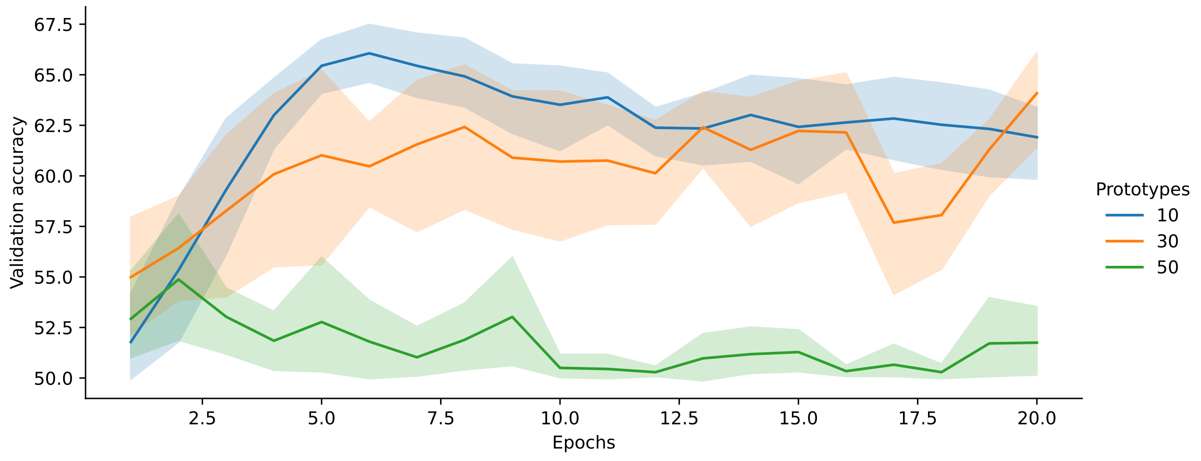

Figure 4.2: Accuracy percent on the validation set by epoch for ProtoryNet models according to their number of prototypes using *leave-some-movies-out* cross-validation. The center line shows the average. The shaded area represents a 95% confidence interval.

such as [27] argue this in a much broader sense.

4.3.3 Challenging the interpretability of ProtoryNet

The key promise of ProtoryNet is its interpretability. Given a sequence of text to be classified, ProtoryNet splits it into sentences, links each sentence to a prototype, and then classifies the text based just on the prototypes related to it. Moreover, each prototype is related to the classes through a score. In our case, scores close to 1 indicate association with the category "female," while scores close to 0 indicate association to the category "male". This should make explanations accessible via the prototypes.

Nevertheless, the trained prototypes of the best performing model (see Table 4.2) and their scores suggest that all prototypes are linked to the category "female" almost to the same degree. With such prototypes one would expect the the model would classify all samples as female. However, as presented in Table 4.1, the model exhibits an accuracy of 66%, and it is better at classifying text from male characters (74% of accuracy vs. 58% for females). Hence, even correct classifications seem unrelated to the prototypes.

Table 4.3 presents some of the dialogues from two characters and the prototypes associated to each sentence (*i.e.*, their *active prototypes*). Both characters were correctly classified by the model with high confidence. Since those examples lie far from the decision boundary, they should be cases on which the explanations and the relationship between the text and prototypes are clear. Nonetheless, it is hard to link each sentence to its active

Prototypes	Score
1. *And what did you do?*	0.72
2. *Osso buco?*	0.74
3. *Oh, you know, the usual small talk, "What's new, how ya been, how do you want to die?"*	0.77
4. *Do you want do you want to try and tell me why you did it?*	0.76
5. *What exactly does that mean?*	0.73
6. *Yes, no, yes.*	0.73
7. *Okay, I gotta tell you.*	0.72
8. *You know, Jack's pretty eager to get up to you know, but, uh, yeah.*	0.64
9. *I have to find some way to get on with my life and I'm going to try.*	0.77
10. *He took a lot of money with him and he didn't come back.*	0.77

Table 4.2: Final prototypes of the best performing ProtoryNet model (with 10 prototypes). The prototype's score indicates the association of the prototype to the classes. A score of 1 (respectively 0) indicates strong association with the category "female" (respectively "male"). Hence, all prototypes show association to the category "female."

prototype. For cases were the character was misclassified, establishing such relationships was equally difficult.

Both, the prototypes' scores and the ambiguous relationship between the text and its active prototypes lead me to question the interpretability of ProtoryNet. It seems that, after mapping each sentence to its active prototype, the model is still taking some decisions which remain occluded in the layers that follow the prototype layer.

Original dialogues	Associated prototype
Female character	
Then why does she act so dumb?	*What exactly does that mean?*
She hasn't got any sense.	*I have to find some way to get on with my life and I'm going to try.*
In some ways, she's awful dumb.	*I have to find some way to get on with my life and I'm going to try.*
Buddy, I'm worried about Rose.	*I have to find some way to get on with my life and I'm going to try.*
No, Mother, he's lying You know, Buddy, sometimes you make me sick.	*Oh, you know, the usual small talk, "What's new, how ya been, how do you want to die?"*
Wasn't Daddy wonderful?	*He took a lot of money with him and he didn't come back.*
He wanted to kiss her some more and play with her, but he didn't, because he loves Mother and all of us, and he loves Rose, too.	*He took a lot of money with him and he didn't come back.*
Isn't he wonderful, isn't he great?	*He took a lot of money with him and he didn't come back.*
Male character	
Run a good-cop-bad-cop.	*Osso buco?*
You want to help me swing it?	*Okay, I gotta tell you.*
Stay smart, Bud.	*Okay, I gotta tell you.*
We build a case.	*Osso buco?*
We play by the rules.	*Osso buco?*
What?	*Okay, I gotta tell you.*
That's the angle Jack was working.	*I have to find some way to get on with my life and I'm going to try.*
Dudley must work for Patchett.	*Osso buco?*

Table 4.3: Sample sentences from two characters and the prototypes associated to each sentence according to the best performing ProtoryNet model. Both characters were correctly classified by the model.

Conclusion

I evaluated the performance of ProtoryNet in a text classification task where we aimed to distinguish the gender of film characters based on their speech. To properly assess its capabilities I compared it to different intrinsically interpretable models (logistic regression and complement naïve bayes) and deep models regarded as black-boxes (fully connected networks, LSTMs, Bidirectional LSTMs and BERT models). To this end, a dataset of film dialogues was assembled based on existing films corpora. Using word embeddings and WEAT, I found moderate (but not statistically significant) gender biases in the dataset.

Among the main results I observe that ProtoryNet's accuracy is comparable to that of well established deep-models such as Bidirectional LSTMs. Nevertheless, in this specific task, classic intrinsic interpretable models, such as logistic regression, offer better accuracy and clearer interpretation. Furthermore, the explanations provided by ProtoryNet's prototypes were ambiguous, questioning the model's self-explainability. It seems that ProtoryNet is still taking decisions that remain occluded in its layers. Defining interpretability as "the degree to which a human can understand the cause of a decision" [14], at least for this task, I would say ProtoryNet offers little to none of it.

On the other hand, logistic regression models' explanations shed some light on patterns that characterise each gender's dialogues. They show a predominance of references to women in men's speech and vice-versa, as well as a preference of female speakers for adverbial and adjectival expressions, interjectional phrases, and polite words; while cursing, noun-based expressions, and lexical fields such as money, sports and work are male-favoured practices. These findings match of the results of previous works [1, 2, 3, 4, 5].

Possible directions for future work include assessing the interpretability of ProtoryNet in a wider variety of datasets, since it appears to perform well in the original work for the task of distinguishing positive and negative reviews. Also, it would be interesting to investigate more thoroughly prototype initialisation schemes since those could be important to the final quality of the prototypes.

Bibliography

[1] Lucia Busso and Gianmarco Vignozzi. Gender Stereotypes in Film Language: A Corpus-Assisted Analysis. pages 71–76. January 2017.

[2] Alexandra Schofield and Leo Mehr. Gender-Distinguishing Features in Film Dialogue. In *Proceedings of the Fifth Workshop on Computational Linguistics for Literature*, pages 32–39, San Diego, California, USA, June 2016. Association for Computational Linguistics.

[3] Apoorv Agarwal, Jiehan Zheng, Shruti Kamath, Sriramkumar Balasubramanian, and Shirin Ann Dey. Key Female Characters in Film Have More to Talk About Besides Men: Automating the Bechdel Test. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 830–840, Denver, Colorado, May 2015. Association for Computational Linguistics.

[4] Anil Ramakrishna, Victor R. Martínez, Nikolaos Malandrakis, Karan Singla, and Shrikanth Narayanan. Linguistic analysis of differences in portrayal of movie characters. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1669–1678, Vancouver, Canada, 2017. Association for Computational Linguistics.

[5] Na Cheng, R. Chandramouli, and K.P. Subbalakshmi. Author gender identification from text. *Digital Investigation*, 8(1):78–88, July 2011. Number: 1.

[6] M. Corney, O. de Vel, A. Anderson, and G. Mohay. Gender-preferential text mining of e-mail discourse. In *18th Annual Computer Security Applications Conference, 2002. Proceedings.*, pages 282–289, Las Vegas, NV, USA, 2002. IEEE Comput. Soc.

[7] Andreas Liesenfeld, Gábor Parti, Yuyin Hsu, and Chu-Ren Huang. Predicting gender and age categories in English conversations using lexical, non-lexical, and turn-taking features. In *Proceedings of the 34th Pacific Asia Conference on Language, Information and Computation*, pages 157–166, Hanoi, Vietnam, October 2020. Association for Computational Linguistics.

[8] Ananya, Nitya Parthasarthi, and Sameer Singh. GenderQuant: Quantifying Mention-Level Genderedness. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2959–2969, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[9] Daniel C. Elton. Self-explaining AI as an Alternative to Interpretable AI. In Ben Goertzel, Aleksandr I. Panov, Alexey Potapov, and Roman Yampolskiy, editors, *Artificial General Intelligence*, pages 95–106, Cham, 2020. Springer International Publishing.

[10] Dat Hong, Stephen S. Baek, and Tong Wang. Interpretable Sequence Classification Via Prototype Trajectory. *CoRR*, abs/2007.01777, 2020. arXiv: 2007.01777.

[11] Cristian Danescu-Niculescu-Mizil and Lillian Lee. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*, 2011.

[12] Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, 2017. _eprint: https://www.science.org/doi/pdf/10.1126/science.aal4230.

[13] Ryan Steed and Aylin Caliskan. Image Representations Learned With Unsupervised Pre-Training Contain Human-like Biases. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 701–713, Virtual Event Canada, March 2021. ACM.

[14] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

[15] Christoph Molnar. *Interpretable Machine Learning*. Lulu.com, 2 edition, 2022.

[16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery. event-place: San Francisco, California, USA.

[17] Damien Garreau and Dina Mardaoui. What does LIME really see in images? In *ICML 2021 - 38th International Conference on Machine Learning*, virtual, United States, July 2021.

[18] David Alvarez-Melis and Tommi S. Jaakkola. Towards Robust Interpretability with Self-Explaining Neural Networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 7786–7795, Red Hook, NY, USA, 2018. Curran Associates Inc. event-place: Montréal, Canada.

[19] Chaofan Chen, Oscar Li, Alina Barnett, Jonathan Su, and Cynthia Rudin. This looks like that: deep learning for interpretable image recognition. *CoRR*, abs/1806.10574, 2018. arXiv: 1806.10574.

[20] Yao Ming, Panpan Xu, Huamin Qu, and Liu Ren. Interpretable and Steerable Sequence Learning via Prototypes. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery &amp; Data Mining*, KDD '19, pages 903–913, New York, NY, USA, 2019. Association for Computing Machinery. event-place: Anchorage, AK, USA.

[21] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015. Publisher: Public Library of Science San Francisco, CA USA.

[22] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

[23] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, October 2017.

[24] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pages 4768–4777, 2017.

[25] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *Workshop at International Conference on Learning Representations*, 2014.

[26] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning Important Features Through Propagating Activation Differences. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153. PMLR, August 2017.

[27] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, May 2019.

[28] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. Publisher: IEEE.

[29] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. Issue: 1.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. Publisher: MIT Press.

[31] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. 2013.

[32] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013.

[33] Anthony G Greenwald, Debbie E McGhee, and Jordan LK Schwartz. Measuring individual differences in implicit cognition: the implicit association test. *Journal of personality and social psychology*, 74(6):1464, 1998. Publisher: American Psychological Association.

[34] Gerard Salton and Chung-Shu Yang. On the specification of term values in automatic indexing. *Journal of documentation*, 1973. Publisher: MCB UP Ltd.

[35] Jason D Rennie, Lawrence Shih, Jaime Teevan, and David R Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 616–623, 2003.

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[37] François Chollet et al. Keras. `https://keras.io`, 2015.

[38] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[39] Kunihiko Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3):121–136, 1975.

[40] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

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

[42] Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 19–27. IEEE Computer Society, 2015.

[43] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal Sentence Encoder. *CoRR*, abs/1803.11175, 2018. arXiv: 1803.11175.

[44] Francesco E. Maranzana. On the location of supply points to minimize transportation costs. *IBM Syst. J.*, 2(2):129–135, 1963.

Appendix

Model	Average test accuracy (%)		
	General	Male	Female
E_{64} + BLSTM + D_{64}^{gelu} + D_{32}^{gelu} + sigmoid	71.37	73.33	69.42
E_{64} + BLSTM + D_{64}^{gelu} + D_{32}^{gelu} + softmax	70.54	80.00	61.16
E_{128} + LSTM + D_{64}^{gelu} + D_{32}^{gelu} + softmax	69.71	79.17	60.33
E_{128} + BLSTM + D_{64}^{relu} + softmax	69.29	65.83	72.73
E_{64} + LSTM + D_{32}^{gelu} + D_{16}^{gelu} + softmax	69.29	65.83	72.73
E_{128} + BLSTM + D_{32}^{gelu} + D_{16}^{gelu} + softmax	68.88	64.17	73.55
E_{128} + BLSTM + D_{32}^{relu} + sigmoid	68.88	63.33	74.38
E_{128} + LSTM + MaxPool + softmax	68.88	70.00	67.77
E_{128} + BLSTM + D_{64}^{gelu} + softmax	68.46	70.83	66.12
E_{64} + BLSTM + D_{32}^{relu} + softmax	68.46	64.17	72.73
E_{128} + MaxPool + D_{64}^{relu} + D_{32}^{relu} + sigmoid	68.46	68.33	69.42
E_{64} + BLSTM + D_{32}^{relu} + sigmoid	68.46	82.50	54.55
E_{64} + BLSTM + D_{32}^{gelu} + D_{16}^{gelu} + sigmoid	68.46	80.00	57.02
E_{64} + BLSTM + D_{32}^{gelu} + D_{16}^{gelu} + softmax	68.46	70.83	66.12
E_{64} + MaxPool + D_{32}^{gelu} + D_{16}^{gelu} + sigmoid	68.05	68.33	67.77
E_{128} + MaxPool + D_{64}^{gelu} + D_{32}^{gelu} + sigmoid	68.05	67.50	68.60
E_{64} + BLSTM + D_{32}^{gelu} + softmax	68.05	68.33	67.77
E_{64} + LSTM + D_{64}^{gelu} + D_{32}^{gelu} + softmax	67.63	58.33	76.86
E_{128} + BLSTM + D_{32}^{relu} + softmax	67.63	69.17	66.12
E_{128} + MaxPool + D_{32}^{relu} + D_{16}^{relu} + sigmoid	67.63	65.00	70.25
E_{64} + BLSTM + D_{64}^{relu} + D_{32}^{relu} + sigmoid	67.63	67.50	67.77
E_{64} + BLSTM + D_{64}^{relu} + D_{32}^{relu} + softmax	67.63	81.67	53.72
E_{128} + BLSTM + D_{64}^{gelu} + D_{32}^{gelu} + sigmoid	67.63	61.67	73.55
E_{128} + LSTM + MaxPool + sigmoid	67.22	83.33	51.24
E_{128} + LSTM + D_{32}^{gelu} + sigmoid	67.22	74.17	60.33

Model	Average test accuracy (%)		
	General	Male	Female
E_{128} + BLSTM + D_{32}^{gelu} + D_{16}^{gelu} + sigmoid	67.22	70.83	63.64
E_{64} + LSTM + D_{64}^{gelu} + sigmoid	66.80	71.67	61.98
E_{64} + LSTM + D_{32}^{relu} + D_{16}^{relu} + softmax	66.80	61.67	71.90
E_{64} + LSTM + D_{64}^{relu} + sigmoid	66.80	71.67	61.98
E_{128} + LSTM + D_{64}^{relu} + D_{32}^{relu} + sigmoid	66.80	62.50	71.07
E_{64} + BLSTM + D_{64}^{relu} + softmax	66.80	65.83	67.77
E_{64} + MaxPool + D_{64}^{relu} + D_{32}^{relu} + sigmoid	66.80	70.00	63.64
E_{128} + MaxPool + D_{32}^{gelu} + D_{16}^{gelu} + softmax	66.80	61.67	71.90
E_{64} + LSTM + D_{32}^{relu} + sigmoid	66.39	82.50	50.41
E_{128} + LSTM + D_{64}^{relu} + sigmoid	66.39	62.50	70.25
E_{128} + LSTM + D_{64}^{gelu} + D_{32}^{gelu} + sigmoid	66.39	65.83	66.94
E_{128} + BLSTM + D_{64}^{gelu} + sigmoid	66.39	82.50	50.41
E_{64} + LSTM + D_{32}^{gelu} + sigmoid	66.39	56.67	76.03
E_{128} + BLSTM + D_{32}^{gelu} + softmax	65.98	78.33	53.72
E_{64} + LSTM + D_{32}^{relu} + D_{16}^{relu} + sigmoid	65.98	85.83	46.28
E_{64} + LSTM + D_{32}^{gelu} + softmax	65.98	64.17	67.77
E_{128} + BLSTM + D_{64}^{relu} + D_{32}^{relu} + sigmoid	65.98	59.17	72.73
E_{128} + LSTM + D_{32}^{relu} + softmax	65.98	72.50	59.50
E_{64} + LSTM + MaxPool + sigmoid	65.98	54.17	77.69
E_{64} + LSTM + D_{64}^{gelu} + softmax	65.56	60.83	70.25
E_{128} + BLSTM + D_{64}^{relu} + sigmoid	65.15	68.33	61.98
E_{128} + LSTM + D_{32}^{gelu} + D_{16}^{gelu} + softmax	65.15	67.50	62.81
E_{128} + LSTM + D_{64}^{gelu} + sigmoid	65.15	62.50	67.77
E_{64} + LSTM + D_{32}^{relu} + softmax	65.15	61.67	68.60
E_{64} + LSTM + D_{64}^{gelu} + D_{32}^{gelu} + sigmoid	65.15	58.33	71.90
E_{64} + BLSTM + D_{64}^{relu} + sigmoid	64.73	59.17	70.25
E_{64} + BLSTM + D_{64}^{gelu} + softmax	64.73	51.67	77.69
E_{128} + LSTM + D_{32}^{gelu} + D_{16}^{gelu} + sigmoid	64.73	69.17	60.33
E_{64} + BLSTM + D_{32}^{relu} + D_{16}^{relu} + softmax	64.73	60.83	68.60
E_{64} + LSTM + D_{32}^{gelu} + D_{16}^{gelu} + sigmoid	64.32	50.83	77.69
E_{64} + BLSTM + D_{64}^{gelu} + sigmoid	64.32	42.50	85.95
E_{128} + MaxPool + D_{32}^{relu} + D_{16}^{relu} + softmax	64.32	69.17	59.50
E_{128} + LSTM + D_{64}^{relu} + D_{32}^{relu} + softmax	64.32	71.67	57.02
E_{128} + MaxPool + D_{64}^{gelu} + D_{32}^{gelu} + softmax	64.32	62.50	66.12

Model	Average test accuracy (%)		
	General	Male	Female
E_{128} + LSTM + D_{32}^{gelu} + softmax	64.32	70.83	57.85
E_{64} + MaxPool + D_{64}^{relu} + D_{32}^{relu} + softmax	63.90	59.17	68.60
E_{64} + MaxPool + D_{64}^{gelu} + D_{32}^{gelu} + softmax	63.90	65.83	61.98
E_{128} + LSTM + D_{32}^{relu} + sigmoid	63.90	60.83	66.94
E_{128} + BLSTM + D_{32}^{gelu} + sigmoid	63.49	59.17	67.77
E_{64} + MaxPool + D_{64}^{gelu} + D_{32}^{gelu} + sigmoid	63.49	55.00	71.90
E_{128} + MaxPool + D_{64}^{relu} + D_{32}^{relu} + softmax	63.07	67.50	58.68
E_{128} + BLSTM + D_{32}^{relu} + D_{16}^{relu} + sigmoid	63.07	50.00	76.03
E_{128} + MaxPool + D_{32}^{gelu} + D_{16}^{gelu} + sigmoid	63.07	65.00	61.16
E_{64} + LSTM + MaxPool + softmax	62.66	40.00	85.12
E_{128} + LSTM + D_{32}^{relu} + D_{16}^{relu} + sigmoid	62.66	46.67	78.51
E_{128} + BLSTM + D_{64}^{gelu} + D_{32}^{gelu} + softmax	62.24	49.17	75.21
E_{128} + LSTM + D_{32}^{relu} + D_{16}^{relu} + softmax	62.24	48.33	76.03
E_{64} + BLSTM + D_{32}^{relu} + D_{16}^{relu} + sigmoid	62.24	45.00	79.34
E_{64} + LSTM + D_{64}^{relu} + softmax	61.83	57.50	66.12
E_{128} + LSTM + D_{64}^{gelu} + softmax	61.41	63.33	59.50
E_{64} + MaxPool + D_{32}^{gelu} + D_{16}^{gelu} + softmax	61.00	61.67	60.33
E_{64} + BLSTM + D_{32}^{gelu} + sigmoid	60.58	57.50	63.64
E_{64} + MaxPool + D_{32}^{relu} + D_{16}^{relu} + softmax	60.58	38.33	82.64
E_{64} + MaxPool + D_{32}^{relu} + D_{16}^{relu} + sigmoid	60.17	60.00	60.33
E_{64} + LSTM + D_{64}^{relu} + D_{32}^{relu} + softmax	60.17	61.67	58.68
E_{128} + LSTM + D_{64}^{relu} + softmax	58.92	51.67	66.12
E_{64} + LSTM + D_{64}^{relu} + D_{32}^{relu} + sigmoid	58.51	52.50	64.46
E_{128} + BLSTM + D_{32}^{relu} + D_{16}^{relu} + softmax	57.68	38.33	76.86
E_{128} + BLSTM + D_{64}^{relu} + D_{32}^{relu} + softmax	53.94	48.33	59.50

Table 4: Deep models' accuracies. E_ℓ denotes an embedding model with output of dimensionality ℓ. D_d^f represents a dense layer with output of dimensionality d and activation function f. The LSTM and Bidirectional LSTM layers (BLSTM) have output dimensionality of 64. Finally, the term "MaxPool" refers to a global max-pooling.

Group	Words
Male	male, man, boy, brother, he, him, his, son
Female	female, woman, girl, sister, she, her, hers, daughter
Male names	John, Paul, Mike, Kevin, Steve, Greg, Jeff, Bill
Female names	Amy, Joan, Lisa, Sarah, Diana, Kate, Ann, Donna
Science	science, technology, physics, chemistry, Einstein, NASA, experiment, astronomy
Art	poetry, art, dance, literature, novel, symphony, drama, sculpture
Career	executive, management, professional, corporation, salary, office, business, career
Family	home, parents, children, family, cousins, marriage, wedding, relatives
Instruments	cello, guitar, clarinet, trumpet, drum, tuba, bell, piano, viola, horn, saxophone, violin
Weapon	whip, mace, arrow, bomb, grenade, missile, rifle, knife, shotgun, gun, pistol, tank
Pleasant	rainbow, laughter, loyal, gift, freedom, caress, diploma, sunrise, love, miracle, honour, vacation
Unpleasant	disaster, abuse, assault, murder, filth, poison, cancer, sickness, prison, ugly, vomit, crash

Table 1: Groups of words used for Word-Embedding Association Tests. The groups "Weapon", "Pleasant", and "Unpleasant" have been randomly shortened to match the size of the group "Instruments" as many words in the latter did not occur frequently enough in the Cornell corpus to figure in the Word2vec embedder.

Model	C	Penalty type	Solver
TF-IDF (unigrams) + LR	0.20	l2	newton-cg
TF-IDF (unigrams, bigrams and trigrams) + LR	0.20	l2	newton-cg
BoW (unigrams) + LR	0.20	l2	saga
BoW (unigrams, bigrams and trigrams) + LR	0.05	l2	sag

Table 2: Best hyperparameters found by the grid search process for logistic regression models. "LR" refers to logistic regression. C is the inverse of regularisation strength parameter.

Model	α	Norm
TF-IDF (unigrams) + CNB	1	False
TF-IDF (unigrams, bigrams and trigrams) + CNB	0.1	True
BoW (unigrams) + CNB	10	False
BoW (unigrams, bigrams and trigrams) + CNB	1	False

Table 3: Best hyperparameters found by the grid search process for CNB models. α is the additive smoothing parameter. The norm parameter decides whether or not a second normalisation of the weights is performed.