Implementing ANNs with Tensorflow WS 2021/22

Universal Language Model Fine-tuning for Fake-Review Classification: Understanding transfer learning with state-of-the-art NLP applications

Tim Petersen, Tim Kapferer, Sofia Worsfold

April 10, 2022

Abstract

As the elaboration of novel techniques in the field of Machine Learning progresses in time, researchers have introduced a Cornucopia of various applications and approaches, centered around the main focus of optimizing pre-existent Machine Learning processes such as artificial neural networks. In a sea of networks where the trend lies in deep layering strategies and harvesting as much labeled data as possible, Howard and Ruder (2018) conceived a text classification application that defies these tendencies: Universal Language Model Fine-tuning for Text Classification (ULMFiT).

In the scope of this paper we would like to inspect the effectiveness of ULMFiT, attempting to shed a light at the concepts that rendered it's success as a malleable Natural Language Processing (NLP) all-rounder. In order to understand the underlying processes of ULMFiT, we propose a reimplementation of Howard and Ruder's paper from 2018. We shall test our network on selected data from the open-sourced 'Fake Review Dataset' of the Open Science Foundation, before sliding into discussions about the real-world consequences of such a powerful instrument.

1 Introduction

If we care to underpin the source of Howard and Ruder's breakthrough in NLP, we shall take a detour in the neighbouring division of Computer Vision (CV).

While Machine Learning algorithms in CV have been surging with increasing accuracy scores on datasets growing even sparser for the past 20 years, NLP has reached comparable success only in the last few years. Many fairly advanced CV applications, such as face recognition have paved their way into our daily lives. [CWW+13] for example, managed to create a robust face recognition algorithm based on a Bayesian model with a simple EM Algorithm in just 2013. To further illustrate, feature recognition and classification in NLP were partly achieved through Support Vector Machines around that time, and thus suffered from very mediocre accuracy and a solid amount of overfitting. [KCD20]

Yet the question arises: what has caused this dissonance? The aims of both fields seem to converge in the way that they care to learn from the 'hidden' underlying structures of what keeps either the visual or linguistic world together, yet many researchers have failed on their attempt to transfer methods from CV to the NLP department. [MMY+16a]

Now, before getting into the details of our text classifier of interest, ULMFiT, we shall further investigate the CV-originated Machine Learning method called 'Transfer Learning', which, as we will see later, greatly inspired the work of Howard and Ruder.

2 Transfer Learning

2.1 Transmitting knowledge in Machine Learning

In a first step, we shall analyse the concept of transfer learning in a broader sense, before we have a look on it's implications for NLP tasks.

The idea of transferring knowledge gained from one task to another task has been pretty much innate within the evolution of us humans. Especially when it comes to our understanding of language, research has shown that we apply remarkable transfer learning skills when learning a new language, to an extent where the number of different languages learned build up to an exponentially growing learning rate. [MMY+16b]

The takeaway message from this phenomenon is the following: the stronger the basis of knowledge we have in a certain domain, the easier (and less costly in resources) it is to solve tasks from a related domain. Figure 1 illustrates these knowledge transfers.

As early as in 1991, Lorien Pratt et. al. expected the benefits of transfer learning methods to be applicable for machine learning algorithms, and proposed one of the first artificial neural networks exhibiting transferred knowledge in a speech recognition task. [PMK91] Antecedent to this event, traditional ML applications trained on isolated tasks. Usually, the training data and testing data were taken from the same domain, in such a way that the input feature space and data distribution characteristics were identical. [MMY+10] As it can become quite laborious, sometimes even impossible, to gather thousands of in-domain data (not to mention the computational expensiveness), Pratt et. al. understood that there must be a way to bypass the tedious work of rewriting every neural network from scratch.

The core idea presented in Pratt et. al.'s paper is that one should know better than randomly initialising the weights and biases when setting up a neural network. Rather, experiments have been conducted where a small neural network, trained on a subtask of the target task, transmitted it's acquired knowledge of so-called hyperplanes (a multi-dimensional representation of the training data, ideally separating the training data such that no decision region contains training data items with different target values) directly towards the main network as a first training step for the latter.

The experiment yielded faster learning in the cases where the network inherited information from it's predecessor than a network trained from scratch. In parallel, transfer learned networks cut the number of necessary epochs before convergence in half for some trials. However, in a few cases, Pratt et. al. report that some cases failed to converge at all, which might be due to a faulty feature learned on the subtask. [PMK91]

2.2 Introducing transfer learning in Computer Vision

After the Machine Learning journal picked up on the topic of transfer learning in 1997, various approaches have been made in every possible section of AI. Computer Vision in particular benefited from this discovery, as the first open-sourced models trained on a variety of image datasets such as ImageNet or MS-COCO emerged.

With pre-trained models being openly available for further use, achieving high performances was no longer optional when being low on resources. Furthermore, this novel accessibility led to a cycle where almost any visual task, such as feature recognition or classification, could be induced from another model, permitting very promising results: sometimes even better than human capacities. The process of adapting a network from a base model is known as 'fine-tuning', which, as we will investigate later in further detail, will be our main component of interest when dealing with ULMFiT later on. [HR18b]

In the meanwhile, an example of what can be achieved nowadays is SpotTune, an algorithm based on ImageNet proposed in 2019 by Guo et. al., which figures out fine-tuning on the classifier by itself. Impressive results show that the network is indeed capable of finding the most (or at least very) suitable parameters for a variety of given tasks, which leaves us with the impression that we are just that much closer to developing self-optimising algorithms in the future. [GSK+19]

On another note, experts foresee an upcoming greater use of transfer learning for commercial use: will the AI industry, that mainly focused on the fruits of large data storage and supervised learning until yet, undergo a turning point on these aspects? [RM19]

2.3 A beginning in Natural Language Processing

What has been discussed until now leaves us with the question: where has this progress been applied in the field of NLP? If we come back to what Howard and Ruder mentioned: it's been stuck somewhere between a lot of failed attempts at reproducing what has been achieved in CV. An explanation for this phenomenon is provided by the two researchers as well: '(...) not the idea of LM [Language Model] fine-tuning but our lack of knowledge of how to train them effectively has been hindering wider adoption.' [HR18b]

It seems that NLP and CV fail to share a certain crucial aspect in terms of transfer learning applicability. Taking an example from CV, it has been considered that convolutional neural networks build up semantics from macro to micro, meaning that general features are processed in the first layers, while the details are captured gradually in the layers preceding the output layer.

In contrast, an NLP network computes semantic relations on a more shallow level, where the context is extracted from the interrelationship between words and their linguistic classes. [MZJ+21] Accordingly, semantic analysis in NLP requires a parameter distribution different from CV applications, gained from the appropriate fine-tuning techniques. One way to illustrate the difference between the two is the concept of hypercolumns. When referring to hypercolumns in CV, we consider the vector composed of all the activations in the layers preceding a certain output pixel in a convolutional neural network, whereas a hypercolumn in NLP depends on the concatenation of word embeddings at different layers with word embeddings obtained from a pre-trained model. [HR18b]

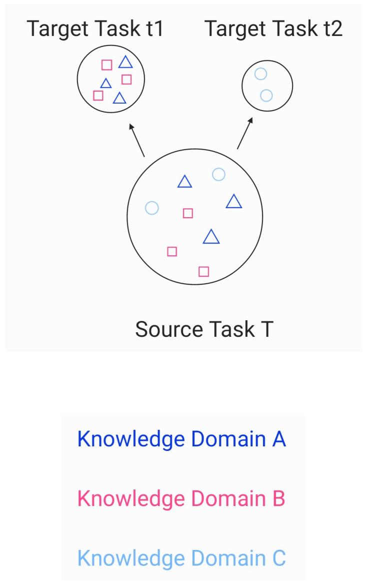

Figure 1: From general-domain to task-specific: Knowledge transmission between a source task T and target tasks t_1/t_2.

3 ULMFiT

Before we present our own attempt at reimplementing ULMFiT, we shall analyse in greater detail the main building blocks of the original implementation from Howard and Ruder in 2018. Please refer to Hence, we will introduce the algorithm by describing the purpose of a general-domain language model, followed by it's respective fine-tuning techniques. We will continue with examining the target task classifier before having a look at analogous work.

3.1 Language modeling

As we demonstrated earlier on the topic of transfer learning, our main conceptual objective is the formulation of a model, which, adapted to and trained on a certain Source task T, will facilitate, both in terms of data and hardware resources, the processing and solving of another subtask t_i.

Inductively, it lays within a researcher's interest to formulate a model capable of *generalising*, whereas we leave the task specific mechanisms in the hands of a *specialising* unit. When trying to apply this reasoning in an NLP environment, we are looking for a function with the ability to synthesize the underlying structure of Natural language in a broad way. This process is named language modeling, and is widely used in many NLP applications such as machine translation or next word prediction.

Now, coming back to ULMFiT, we observe that Howard and Ruder expect a universal aspect to their language model, thus the first three letters of the acronym. In this subsection, we shall analyse the language model of ULMFiT in terms of what it is composed of, and, in a second part, we will discuss how it achieves to be universal as promised by Howard and Ruder.

3.1.1 AWD-LSTM and the statistical model

Starting with the atoms of our language model: the AWD-LSTM, short for ASGD Weight Dropped on the left hand side and Long Short Term Memory on the right hand side. To choose this special kind of recurrent neural network structure has the benefits, as the name specifies it, of disposing of an awareness of long and short term dependencies. This is crucial to the understanding of language, since we make sense of a word in relation to where it stands in a specific sentence. In a mathematical way, we can say that our language model is a statistical model, constructed around a distribution of conditional probabilities for each word in a sentence. Considering a set of example words (or chunks of sentences) $(x_1, x_2, ..., x_n)$ we would like to determine $p(x_n|x_1, x_2, ..., x_{n-1})$. [RWC+19] Ideally, our model should be able to predict the next word in a sentence, e.g. figure out which word is more probable to occur at this point of the sentence given the context. Perplexity is a metric defined to evaluate the quality of a language model, where low perplexity indicates a senseful estimation of the conditional probabilities. Let's take an example sentence 'The sun is in the' where we would like to predict the next word. As in this case it should be rather unchallenging for any minor model to compute 'sky', we expect low perplexity. Our model is not 'perplexed' when seing the word 'sky' at the end of the sentence. In contrast, let's consider the sentence 'I've spent several years in China. I speak fluent'. Here, our model is confronted with context being laid out further back in time, we expect the perplexity to be higher. [CBR98]

Coming back to the AWD-LSTM and it's properties, we should mention the particular structure it utilises to infer a sense of time. An LSTM cell manages to pass on information, through means of various flow-controlling gates, from one point in time to a later one, information that is then concatenated, or not, with the input in the current state. AWD defines a dropout policy for performance boosting. In summary, training the language model results in finding a function that adequately formulates the long and short term dependencies between words in a sentence or text, on an unsupervised basis. As a consequence, the model should show the ability to perceive the grammatical hierarchy between words, as well as, as shown by Radford et. al in 2017, an understanding for different levels of sentiment (positive, negative, or neutral) that could affect the context of a sentence. [RJS17]

3.1.2 Word embeddings and tokenization

For the language model to be able to establish a relationship between words, we must think of a suitable representation for the words we are going to feed into our network. We make use of so called embeddings to map the variety of language onto real-valued numbers. In turn, the vectorisation of language permits the computation of similarity between the semantics of different words. Howard and Ruder used an embedding size of 400, meaning that each word or sentence piece inherits a 400-dimensional vector space. [SLMJ15]

Further encoding takes place during the process of tokenization, where the texts will be split in tokens (e. g. chunks of sentences) in order to construct a vocabulary consisting of token-integer pairs. In one forward pass through the language model during training, the tokenized input sentence receives it's respective embedding through the embedding layer before it gets fed into the AWD-LSTM. With the help of an embedding matrix and one-hot encoding, attributing the right embedding is a simple matter of matrix multiplication. After traversing the 3 LSTM layers, a softamx decoding layer permits the output of probability distributions for the next word prediction. [HR18a]

3.1.3 How is generalisation affecting competence?

Howard and Ruder opted for the Wikitext-103 dataset, which includes almost 30,000 different Wikipedia articles that have been labeled as 'good' by the community. The reported state-of-the-art results after full training of their complete network demonstrates that the language model must have performed well at generalising given the data. However, Howard and Ruder expect even higher performances for future models: 'We leave the exploration of more diverse pretraining corpora to future work, but expect that they would boost performance.' [HR18b]

For the sake of completion, we shall investigate what the future has in fact brought upon us concerning this prediction. We hence prepare to answer the following questions: (1) Does a larger dataset improve performance on the language model? (2) Does a language model perform better when extending the number of hyperparameters? To be noted, we discard any information about quality in our questioning, since evaluating a consequence of quantity allows better tractability.

In order to answer our first question, we consider the work of Blinov et. al from 2019, which demonstrated how ULMFiT can be adapted to classifying humoristic expressions from non-humoristic ones. The resulting ULM-Fun model was then used to explore the relationship between dataset size and model performance. In summary, the language model was pre-trained on a corpus of 10 Million internet forum entries for 15 epochs, as well as on a self-composed dataset (baptised as the FUN dataset) varying in size for the fine-tuning process for 5 epochs. The architecture of the network was directly imported from the original ULMFiT implementation.

As shown in Figure 2., results indicate that a larger dataset induces a higher F1-score, which in turn proves that the network performs better at classifying when blessed with a larger dataset. [BBBB19]

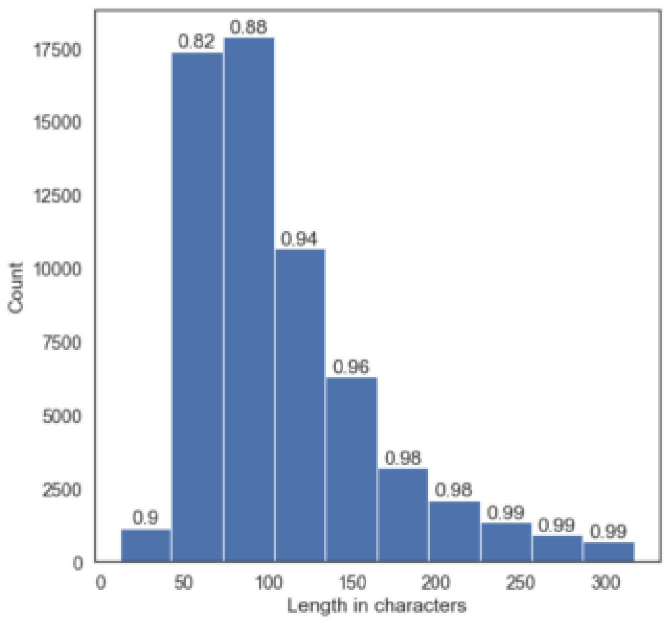

Figure 2: F1-scores for humor class depending on text length in FUN test set. Taken from [BBBB19]

While preferences in research shifted more towards a self-attention architecture using transformer-based models rather than LSTMs, it is still worthy to mention their contribution in language modeling in order to answer question (2). Roberts et. al for instance took on research about how the number of hyperparameters in their model affects productivity. They made use of the T5, the text-to-text transformer, which is notably different from our model of interest, but still represents the concept of a knowledge base within the language model to a similar extent. The experiment relied on a relatively trivial question answering task, although knowledge retrieval operated solely via internal processes, in order to capture the knowledge scope of the model without external aid. The model was trained on a set of queries accompanied by an url that, indirectly, gives the answer to the respective query. During the process of research, 5 different models, varying in the number of hyperparameters from 220 Million to 11 Billion, were trained on the same dataset with identic conditions. Results yield that the largest model does in fact account for higher recall scores, which demonstrates the ability of the model to generalise even when given more parameters to play with. However, Roberts et. al do point out that the degree of difficulty related to the trivia-style question and answer system remains low and hence does not represent to what extent the model is capable of reasoning, we will retain that model size and knowledge capacity go hand in hand. [RRS20]

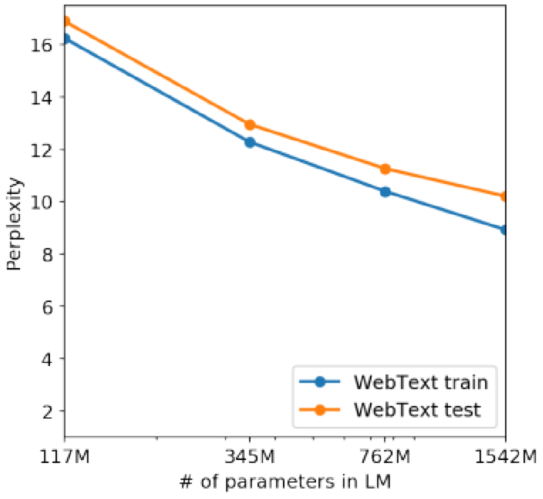

Figure 3: Performance on the Winograd Schema Challenge as a function of model capacity.Taken from [RWC+19]

On another similar attempt to synthesize the impact of larger transformer-based models on performance, we may have a look at Radford et. al's contribution in understanding the OpenAI originated GPT-2 model. Parallel to our first example, 4 models extended with 117 Million to 1542 Million parameters were tested on various NLP tasks, such as question answering and reading comprehension. The model was trained on the WebText dataset, a semi-manually picked set of parsed html files from the internet. Afterwards, the model was tested on the Winograd Schema Challenge, a machine learning problem that requires the model to show reasoning capabilities in order to disentangle the semantic ambiguities occurring in a sentence. As visualised in Figure 3., perplexity diminishes with model growth, which inspires to accept our earlier hypothesis as well. [RWC+19]

3.2 Fine-tuning

In this subsection, we will dive deeper into the task-oriented part of transfer learning. We are now interested in how we can possibly mold our previously conceived language model into something that can work with the desired target task. In order to do so, we will inspect different techniques, some of them introduced by Howard and Ruder themselves, that mostly affect the architecture of the final network in a way that bends the flow of information towards where it can be used efficiently. In the years preceding the publication of ULMFiT, fine-tuning has been applied to several NLP applications, but without greater success and still very dependent on in-domain documents. The main issue was related to forgetting: previously acquired knowledge, which could be crucial for further analysis, might be discarded wrongfully during the process of training; not to mention other pathologies such as overfitting when acquainted with a shortage of data. [HR18b]

In describing the fine-tuning procedures applied to the language model in a first part, as well as the classifier in a second part, we aspire to demonstrate the sophisticated engineering of ULMFiT.

3.2.1 Language model fine-tuning

The process of fine-tuning the language model is needed for adjusting the general-domain documents to the in-domain documents of our target task, since both domains very likely won't overlap. Apart from some minor changes of size in the output layer, the architecture remains the same as computed during pre-training of the language model. Changes are mainly going to be made on the hyperparameter side of the network. Howard and Ruder put an accent on the importance of choosing the right hyperparameter for a certain layer, since each layer captures different facets of the input language. In an effort to apply this theory, two learning rate distribution techniques are introduced: *discriminative fine-tuning* and *slanted triangular learning rates*.

Discriminative fine-tuning installs a hierarchy between the learning rates of each layer. In general, the learning rate increases with each layer, which translates to taking bigger steps during the gradient descent as information progresses through the network. In more detail, as empirically stated, performance peaks when the learning rate is divided by 2.6 starting from the last layer downwards.

The mechanism underlying the slanted triangular learning rates method can be visualised as a function between learning rates η and iteration number i, where η first linearly increases with a slope of approximately 4, then, after $i = 200$, the slope decreases with a factor of approximately -0.7. This method has shown to be more effective than previous approaches, such as annealing learning rates, for the reason that it incites the model's parameters to converge to a suitable subspace rather quickly, before getting exposed to minor changes (fine-tuning, again) stretched over a larger period of time. [HR18b]

3.2.2 Target task classifier

As a final step in constructing ULMFiT, we need to integrate the actual classifying unit. To do so, we extend the model we have built up until now with two extra linear feed-forward neural network blocks, where the first one is driven by a ReLU activation function and the second manages probability distributions using softmax. For the sake of fighting the forgetting syndrome, Howard an Ruder apply concat pooling of previous layers before feeding them into the classifying unit. This permits integrating information gained in previous layers without focusing solely on the last layer. In summary, concat pooling happens when the last hidden LSTM state is concatenated with a max-pooled and a mean-pooled representation of the hidden states.

Notably, the classifier is the only element in the algorithm that is trained from scratch. As it is also initialised with random weights and biases, it makes sense to train the last layer before training the rest of the network. This is where gradual unfreezing comes in handy. As the name suggests it, only the last layer is trained in the first epoch, while the rest is 'frozen'. As the ice unfreezes, e.g. we traverse more epochs, the remaining layers in turn get trained. [7]

3.3 Related work

As we've already been mentioning in the part where we analysed the competence of language models, the usage of LSTM-based networks in NLP has lost in popularity, whereas transformer-based models such as GPT-2 or the T5 awaken new scientific interest. Hence, we will dedicate this subsection to an off-path excursion towards transformers and their specificities.

In a recent paper published by Gillioz et al., two major issues were connected to the recurrent properties of LSTMs: on the one hand, a purely recurrent network fails to multi-task in the sense where it can't process multiple workflows in parallel, on the other hand, it's still not completely immune to the vanishing gradient problem when confronted with large clauses. To counteract, the transformer utilises a specific attention mechanism, which allows for more tractability concerning the dependencies within language without being tied to a representation of distance. On an outward perspective, the transformer stacks encoding and decoding units to pass on word embeddings, with each unit benefiting from a so-called self-attention layer. This system provides a numeric value for each word in a sequence of words in order to quantify the relatedness from one word to another, therefore granting more expressiveness when concerned with conditional probabilities of words. [18]

BERT, a popular Google application for NLP tasks such as machine translation or reasoning, positions as one of the top transformer-based models today. As previous models processed linguistic sequences in a unidirectional fashion (e.g. from left to right), BERT allows a bidirectional convoy of information which in turn permits a wider understanding of context within a text. Furthermore, adaptation to downstream tasks is conceivable with little effort in fine-tuning. A variety of BERT-based algorithms have emerged since then, such as the sciBert, fine-tuned on a corpus of scientific papers, or the camemBERT, a high-performance BERT trained on french language. [18]

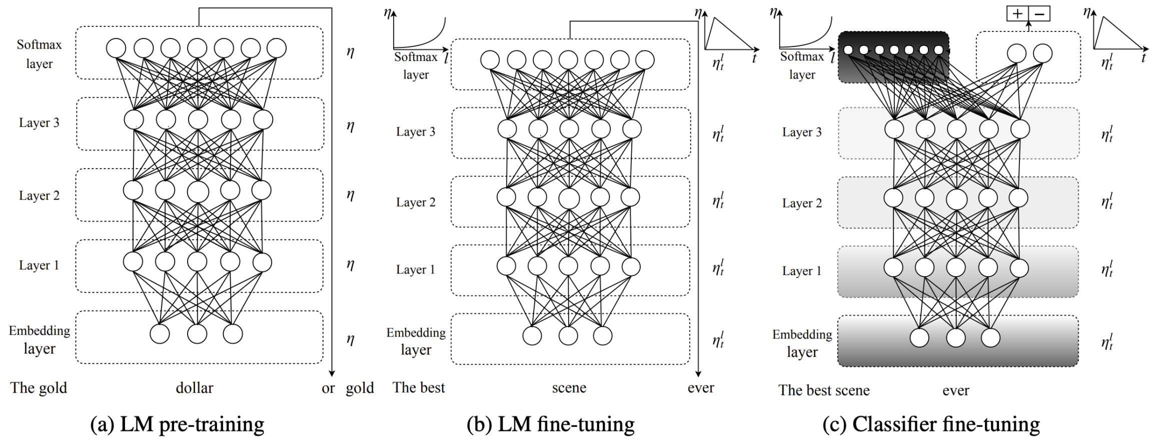

Figure 4: The three stages of ULMFiT: (a) language model pre-training, (b) language model fine-tuning, (c) Classifier fine-tuning. Increasing shading indicates increasing freezing. Taken from [HR18b]

4 ULMFiT Experiment

Finally, we complete our journey with ULMFiT by proposing an implementation of the latter for the purpose of online review classification. While we tried to opt for an implementation that remains identical to Howard and Ruder's, we've encountered few situations where we deviate from the original. We will introduce this section with some background information on the topic of fake online reviews, and why we believe in the importance of detecting them. Afterwards, we will give an insight on our experimental setup, where we invite to visit the respective GitHub repository here for an in-detail description of further technicalities. Closing off, we publish our obtained performances and offer discussion on the results.

4.1 Motivation

Especially during the on-going pandemic of 2020, world-wide economics have experienced a shift from real-world based transactions to a majority of online held transactions. In parallel, the trend towards online shopping doesn't seem to expect notable drawbacks any time soon. Considering the scope of monetary resources gained from a massively growing online clientele, companies strive to expand their online presence by adapting their marketing strategies to the digital world. One way to affect a potential customer into buying a specific product (and arguably one of the most effective way) is through means of user reviews. Ideally, companies rely on positive customer experience to advertise their products. Oppositely, a negative review which might point out on the deficits of a certain item results in abstinence, which rather comes to advantage of the consumer that might have saved his money from being wasted on a seemingly bad product.

We can see here that the interests of both parties, consumer and manufacturer, might not always converge. Where the customer seeks honesty, some companies might resort to different operations. For instance, many of us are familiar to the term 'fake news' and are able to recall the impact of untruthful information in the political sphere, but what about the economical sphere? What if online reviews could simply be generated to impose a bias, negative or positive, on the targeted product? Indeed, history has shown that where lies money there lies power, and somewhere between the two emerges the itching temptation to control both of them. We've discussed how modern NLP applications can achieve human-like linguistic skills, and in this section we shall demonstrate how these technological innovations can be used to make profit in the worst way possible. With the ambition to shed light on the truth, we propose ULMFake: A reimplementation of ULMFiT by Howard and Ruder, fine-tuned to distinguish real online reviews from fake ones.

4.2 Experiment setup

As a first step, we imported a pre-trained language model from edrone, which has been trained on a Wikipedia dataset similar enough to the dataset Howard and Ruder used. The model was originally implemented using the PyTorch framework and fastai library, but with respect to the intent of our course, we used the tensorflow implentation of the latter. At the end, we obtained a vocabulary of size 35,000.

For the hyperparameters, we stayed true to the original and managed learning rates via discriminative fine-tuning and slanted triangular learning rates. For the fine-tuning steps, we trained our model on the in-domain documents: the tensorflow Amazon review dataset. The dataset is composed of reviews from the department of apparel, beauty, books, home, video, and wireless. We hoped to familiarise our model with the syntactics and semantics of the internet world, presuming that the language of the average human being is formulated differently from the language used in a Wikipedia article. Since by now we would have a very large vocabulary size beyond our computational capacity, we decided to only take into account the tokens that intersect with the Amazon review dataset. To that we added 2000 tokens we obtained from training a new sentencepiece model on the Amazon review dataset, resulting in a final size of 5269. As for the classifier, we used a Dense layer with sigmoid activation to output the probability of the review being fake/computer generated, gradual unfreezing included during training. The classifier is then trained on the OSF fake review dataset, composed of labeled examples of either real reviews or fake, computer-generated reviews. Both units (LM fine-tuning and classifier) were trained for 20 epochs each. We display our results in accuracy and loss for training and validation, for both the complete fine-tuned model and the classifier only.

4.3 Results

Training	Validation
'loss: 0.1526785343885422'	'loss: 0.1939335316419601'
'acc: 0.9400475025177002'	'acc: 0.9301543235778809'

Table 1: Training and validation scores of the Classifier built on a fine-tuned language model.

Training	Validation
'loss: 0.261987417936325'	'loss: 0.146187677979469'
'acc: 0.886426568031311'	'acc: 0.9417293071746826'

Table 2: Training and validation scores of the Classifier when applied without domain fine-tuning.

5 Conclusion

In the results we can see that our model performs well on the dataset for training and validation in either case. Though we can see that we reach significant accuracies for training and validation even in a few epochs each. The interesting thing to see is that the model with domain-specific fine-tuned language model was not performing better on our task, than the model without.
A possible reason for that could be, that since our task was not very limited to a specific niche, such as "movie reviews", but rather to reviews in general, we could possibly take advantage of the already broadly trained language model on the Wikipedia articles. Further ideas for following research/projects could be:

- Comparing it to SOTA Transformers in the same subject
- Trying on different datasets and comparing performances
- Scrape different review sources and analyze how many reviews are "fake"

References

[BBBB19] Vladislav Blinov, Valeria Bolotova-Baranova, and Pavel Braslavski. Large dataset and language model fun-tuning for humor recognition. 2019.

[CBR98] SF Chen, D Beeferman, and R Rosenfeld. Evaluation metrics for language models. 1998.

[CWW+13] Xudong Cao, David Wipf, Fang Wen, Genquan Duan, and Jian Sun. A practical transfer learning algorithm for face verification. *Conference on Computer Vision*, 2013.

[GSK+19] Yunhui Guo, Honghui Shi, Abhishek Kumarand, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: Transfer learning through adaptive fine-tuning. 2019.

[HR18a] Jeremy Howard and Sebastian Ruder. fast.ai ulmfit implementation. 2018.

[HR18b] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. 2018.

[KCD20] Abhishek Kumar, Jyotir Moy Chatterjee, and Vicente García Díaz. A novel hybrid approach of svm combined with nlp and probabilistic neural network for email phishing. *International Journal of Electrical and Computer Engineering*, 2020.

[MMY+10] Lili Mou, Zhao Meng, Rui Yan, Ge Liand Yan Xu andLu Zhang, and Zhi Jin. Lisa torrey and jude shavlik. 2010.

[MMY+16a] Lili Mou, Zhao Meng, Rui Yan, Ge Liand Yan Xu andLu Zhang, and Zhi Jin. How transferable are neural networks in nlp applications? 2016.

[MMY+16b] Lili Mou, Zhao Meng, Rui Yan, Ge Liand Yan Xu andLu Zhang, and Zhi Jin. Second language transfer learning in humans and machines using image supervision? 2016.

[MZJ+21] Dastan Hussen Maulud, Subhi R. M. Zeebaree, Karwan Jacksi, Mohammed A. Mohammed Sadeeq, and Karzan Hussein Sharif. State of art for semantic analysis of natural language processing. 2021.

[PMK91] Lorien Y. Pratt, Jack Mostow, and Candace A. Kamm. Direct transfer of learned information among neural networks. 1991.

[RJS17] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. 2017.

[RM19] Ricardo Ribani and Mauricio Marengoni. A survey of transfer learning for convolutional neural networks. 2019.

[RRS20] Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? 2020.

[RWC$^+$19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[SLMJ15] Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. Evaluation methods for unsupervised word embeddings. 2015.