Abstract

1. Introduction

In the last two years transfer learning has developed massively in the NLP field introducing fine tuning approaches akin to those seen in computer vision some years earlier [15]. This growth did not originate from nothing, feature based transfer learning in the form of word embeddings has been in use for some years, particularly driven by [4], and indeed as part of this new wave we have seen advancements in feature based transfer learning in the form of ELMo [12]. However a characteristic trend in this most recent wave of transfer learning models is a class of algorithms that primarily focus on a fine tuning approach, where a base language model [5] is trained, and then is fine tuned on a target task. The fine tuning methodology typically relies on two physical components: a base language model, and an appended part specific to the target task. This base language model is typically very large (100M+ parameters) and takes a long time to train. However the fine tuning task is usually much quicker to train as only a few parameters are added to the model, typically a single dense layer to the end of a multilayer LSTM or transformer. Then the model continues training either all or part of the net but typically on much less data and for much less time as only the task specific information is being learned and the general understanding of language is transfered.

These approaches have on multiple occasions broke the SOTAs across the board on a range of NLP tasks and datasets [2][3]. However all of these datasets are designed for deep learning, they are typically large enough that they warrant the use of deep learning without the necessity of transfer learning.What transfer learning does in these cases is push the boundaries of performance.

At these data scales it has been shown [6] that at these scales deep learning surpasses the limits of classical machine learning algorithms in NLP tasks. So it is natural that the competitive models outperformed by the new transfer learning approaches were all deep learning approaches.

However there exists another use case for transfer learning in NLP, one of particular interest to companies working on real world data; the ability to perform low shot transfer learning. Low shot transfer learning (also referred to as “few-show”) is defined as using transfer learning to train models where we have little training data available. This is of extreme importance as many potential real world applications of machine learning NLP do not have access to sufficiently large datasets to train deep learning algorithms, and getting the data can often be too expensive or time consuming.

Most of the literature on deep transfer learning mention as a corollary point that the approach can be used with low quantities of data to give strong results. However in sources such as [2] results on low-shot learning are presented relative to training deep models from scratch, but deep learning is not the best paradigm at these scales, in [17] they showed that even at scales of ~2000+ labels per class a SVM outperforms several deep learning paradigms on deep learning tasks. As such we propose that to evaluate low shot learning benefits of deep transfer learning models we should in fact look at performance against the strongest classical machine learning methods. However we have yet to find a comprehensive quantitative study performing this analysis and show that low shot transfer learning in NLP is actually the optimal approach when dealing with small quantities of data, or if classical machine learning algorithms would produce comparable or superior results.

In this paper we will attempt to answer this question in the context of classification tasks. What is the best approach to use in the case where we have 100-1000 labeled training examples per class.

The choice of 100-1000 is not arbitrary, these quantities represent the amount of data it is feasible for companies and researchers to tag in house, or amounts of data that occurs organically through other means. For example in marketing they typically represent the base sizes of surveys that can be used as training data for predictors.

The rest of this paper is laid out as followed. Section 2 details the datasets we use. In section 3 we present the algorithms we use to test along with related work influencing our choices in selecting those models. Section 4 looks at the methodology used to evaluate the optimal paradigm. Section 5 details our experiments including choosing the optimal configuration of hyperparameters and preprocessing for each algorithm. Section 6 we present the results, followed by a conclusion in section 7, and discussion of the findings in section 8.

2. Datasets

We have tried to source a range of publically available datasets for classification tasks to remove any potentially unknown biases of one particular set. However to aid in our goal of viewing cross dataset and cross domain performance we have focused on sentiment based classification. This is generally considered to be one of the harder text classification problems [10] as well as being one of critical importance for many companies. Potential biases include the ability for a certain task to be predictable off of a few low level features making the task more trivial, or similar data could have been used in the pre-training of the deep transfer learning approaches tainting the test set.

The datasets we consider fall into two domains: Amazon reviews, and Twitter. The first category consists of 3 Amazon datasets, one consisting of movie reviews, and two of product reviews from different product categories. Whilst these are very real world datasets we describe this domain as clean data. These datasets typically have similar medium length documents of 100 words (see figure 1) and are the kinds of datasets typically used in evaluating the performance of deep transfer learning: [2] uses IMDB and [3] uses SST-2 movie reviews.

The second domain has datasets sourced from Twitter, a social data source that differ in a few key properties from the Amazon sets. Their vocabulary is much broader given the amount of slang, abbreviations, and range of topics discussed. They also use emoji much more than other domains. This can make them much harder to classify, particularly for deep transfer learning models that have pre-defined vocabularies. BERT relies on WordPiece embeddings which makes it more robust to new vocabularies [11], although it still will not handle emoji. On the other hand ULMFiT relies on a set word token vocabulary defined by their training set, which by default is wikitext-103, a wikipedia based text, so this will struggle both with emoji and new vocabulary. We hypothesise these models will suffer a greater loss in accuracy on these Twitter datasets than the classical algorithms because of this fixed vocabulary limitation.

2.1. Amazon Movies

Amazon movies sourced from [7] is a huge collection of movie reviews from Amazon, it includes reviews made up to October 2012. We use a random subset of this for our purposes. All reviews here are on a 5 point scale which we resample to a 3 point scale by binning 2 star with 1 star and 4 sar with 5 star reviews. This is a procedure we use throughout this work to align all of our datasets onto a 3 point sentiment scale of negative, neutral, and positive. In this dataset we also have knowledge of which product each review belongs to so we split out the train, validation, and test sets so no product appears in two or more sets.

2.2. Amazon Books

The second dataset we consider is from the Amazon product review database [8], and contains reviews based on books. This dataset was chosen as it is fairly similar to that of the Amazon movies whilst still being a different domain. This makes it an ideal in helping us avoid biases of the specificity of one dataset whilst training a very similar task. It’s similarity also makes it a perfect candidate to test how well classifiers perform cross domain in a best case scenario.

The dataset is structured similar to that of the movies, with a star rating from 1 to 5 giving us a 5 point sentiment scale which we resample to 3 point. A text review column of document sizes similar to that of the movies as shown in figure 1. It also contains information about which product (in this case book) is being reviewed so we can insure there is no information leakage into the test set by ensuring every book in each set is exclusive.

2.3. Amazon Health and Personal Care

This dataset is almost equivalent to the above in terms of setup however now the reviews are looking at health and beauty products.

2.4. SemEval 2017 Subtask A

SemEval 2017 Task A [13] comes pre tagged into negative, positive, and neutral classes so no binning is necessary. This is the only dataset we use that doesn’t contain “product” information so we simply randomly divide the Tweets between the train, validation, and test sets.

2.5. SemEval 2017 Subtask CE

Again looking at SemEval 2017 [13] we use their subtask CE datasets to produce this set. The data here comes pre-split although as we have a slightly unusual case here wanting only a specific amount of training data and more test data we shuffle everything and resplit. Here we have product information as all tweets are labelled with a topic, so we divide on that variable. Tweets here are also on a 5 point scale but again we bin to a 3 point grouping “very negative” and “negative”, and “very positive” and “positive”.

*Figure 1*

Document lengths

3. Related Work

We are trying to compare the best in class approaches from classical machine learning and fine tuning transfer learning, as such we have tried to leverage models used in other well referenced work, the ones considered in this paper are introduced below.

3.1. Classical ML

However, there is no single classifier that generally achieves the best classification performance. Dinakar et al. [22] showed that NB outperformed DT in their experiment; Davidson et al. [21] found that LR and SVM tended to perform significantly better than other classifiers while Dadvar et al. 23] have shown the NB is slightly better than SVM. As such we have considered two of these to give a fair representation and alleviate the bias of a single classifier.

3.1.1. Naive Bayes

The Naive Bayes is a probabilistic classical machine learning classification algorithm that has a long history of being used in text classification tasks including sentiment analysis. It is well suited to this task given its speed to run and ability to easily incorporate many features [16] which often occurs with classical NLP approaches.

Describe a little about NB

However as with most algorithms from classical machine learning they are not competitive on large datasets compared to deep learning models as such we struggled to find an undisputed best in class Naive Bayes approach. We decided to follow the well referenced approach in [1] as in the paper they clearly show the benefits of each modification they make which we are able to verify for our data in section 5, they also ran their classifier on very a very similar dataset to the ones used in this paper (movie sentiment classification).

3.1.2. SVM

[1]

3.2. Fine Tuning Deep Transfer Learning

3.2.1. ULMFiT

ULMFiT was introduced by [2] and was one of the first popular applications of fine tuning transfer learning in NLP. They achieved SOTAs on various classification datasets: AG, DBpedia, Yelp-bi, and Yelp-full.

Their approach is to use an AWD\_LSTM, an architecture originating in the work of [19] as the language model at the core of the model. The model is then trained in 3 stages, first the core language model is trained on a large general purpose corpus. This is the pre-training step which is ideally done only once per language. Stage 2 is the fine funing step where the same core language model continues to be trained but now on the target dataset, this aids the model in learning the nuances of the model which ultimately improves results on the final classification. The training of the classification task is the third stage, here two dense layers are appended to the final hidden layer of the language model and the whole model is trained now on supervised classification. Advanced techniques such as slanted triangular learning rates and chain-thaw are used to negate the problem of catastrophic forgetting and allow the model to retain earlier learnt information.

Although the model is agnostic to use any pre-training dataset the current published model was built on wikitext-103, however this is actually not very general purpose for many real world applications such as social datasets. The model is also built using a fixed vocabulary which further limits its generalizability.

We use this pre-trained AWD-LSTM based model and follow the recommended fine tuning stages in this paper, whilst verify the choice of all hyperparameters on held out validation sets in section 5.

3.2.2. BERT

BERT (Bidirectional Encoder Representations from Transformers) was introduced in late 2018 by [3]. Conceptually it is similar to that of ULMFiT, a core language model trained on a large general purpose corpus followed by a stage of task specific fine tuning to learn a supervised task. BERT has the generalizability to work with classification or sequence based target tasks which gives it further utility, however in this paper we focus on its ability in classification tasks.

BERT broke various SOTAs on NLP tasks both in classification and other challenges such as question answering.

The pre-training of BERT is more elaborate than that of ULMFiT as they use a fully bidirectional language model where all weights are trained in both directions, they also use random masking, and perform sentence prediction all of which aid in the overall performance of the model. The feature representation is based on word pieces [11] which should allow the model better generalizability than ULMFiT although the vocabulary is smaller at only 30,000 tokens.

On release BERT was published with two models: BERT-base which uses 12 transformer layers and has 110M parameters, and BERT-large with 24 layers and 340M parameters. In this paper we use BERT-base and follow the suggestions of fine tuning as given in the original paper. We verify our model in section 5 on our validation sets.

The second and final stage of training BERT for classification tasks is to append a single dense layer to the final hidden layer of the language model and continue training.

4. Methodology

To answer the question as to what is the optimal approach to use in a classification task when we have 100-1000 labelled examples per class we will compare the performance of a selected set models from classical machine learning and deep transfer learning on various low shot datasets. We are using two approaches from classical and two from transfer learning to try and minimize any bias from one specific model as out goal is to evaluate the approach in general, even though we are limited to contemporary models; for the same reason we shall use a range of datasets.

By performance we shall look at the standard metrics on a held out test set (f1-micro-score). In addition we shall also test the model’s robustness by examining how they perform on highly unbalanced data, on cross dataset predictions, and cross domain predictions. Here we define a domain as a set of datasets that share similar properties, in this paper we consider two domains: Twitter, and Amazon reviews. Although in academic literature we do see attempts at cross domain algorithms [18], the evaluation of algorithms naively on other domains is not often reported. However we see this as a common practice in business to train a single classifier and use it cross domain, and as such we feel it should be evaluated here since our end use case here is informing the opinion on what approaches to take when building real world classifiers. We shall also consider several levels of low-shot transfer learning, taking 100, 300, and 1000 labelled examples per class in a grid search attempt to see if there are any performance cross-over points.

For every dataset we set aside controlled test and validation sets. All fine tuning and hyperparameter selection is done on the validation set and all values displayed in this paper in section 5 are from the held out test sets. The test sets are comprised of products that are independent of the train and validation sets, the definition of a product is specific to each dataset, for example in the Amazon movies dataset a product is a specific film. We do this to keep the test as fair as possible, and ensure that if classifiers overfit and learn features such as the name of the film as a defining feature in sentiment classification it is not rewarded in the evaluation. A summary of the datasets used are shown in table 1. For the unbalanced data we take the mean number of documents per class.

*Table 1*

|  |  |  |
| --- | --- | --- |
|  | Train | Test |
| Amazon Movies Balanced 100 | Positive:100; Neutral:100; Negative:100 | Positive:5000; Neutral:5000; Negative:5000 |
| Amazon Movies Balanced 300 | Positive:300; Neutral:300; Negative:300 |
| Amazon Movies Balanced 1000 | Positive:1000; Neutral:1000; Negative:1000 |
| Amazon Movies Unbalanced 100 | Positive:218; Neutral:34; Negative:48 | Positive:11372; Neutral:1518; Negative:2110 |
| Amazon Movies Unbalanced 300 | Positive:699; Neutral:85; Negative:116 |
| Amazon Movies Unbalanced 1000 | Positive:2229; Neutral:302; Negative:469 |
| Amazon Books Balanced 100 | Positive:100; Neutral:100; Negative:100 | Positive:5000; Neutral:5000; Negative:5000 |
| Amazon Books Balanced 300 | Positive:300; Neutral:300; Negative:300 |
| Amazon Books Balanced 1000 | Positive:1000; Neutral:1000; Negative:1000 |
| Amazon Books Unbalanced 100 | Positive:222; Neutral:45; Negative:33 | Positive:11756; Neutral:1689; Negative:1555 |
| Amazon Books Unbalanced 300 | Positive:707; Neutral:111; Negative:82 |
| Amazon Books Unbalanced 1000 | Positive:2342; Neutral:347; Negative:311 |
| Amazon Health and Beauty Balanced 100 | Positive:100; Neutral:100; Negative:100 | Positive:5000; Neutral:5000; Negative:5000 |
| Amazon Health and Beauty Balanced 300 | Positive:300; Neutral:300; Negative:300 |
| Amazon Health and Beauty Balanced 1000 | Positive:1000; Neutral:1000; Negative:1000 |
| Amazon Health and Beauty Unbalanced 100 | Positive:231; Neutral:31; Negative:38 | Positive:12050; Neutral:1410; Negative:1540 |
| Amazon Health and Beauty Unbalanced 300 | Positive:724; Neutral:87; Negative:89 |
| Amazon Health and Beauty Unbalanced 1000 | Positive:2396; Neutral:294; Negative:310 |
| SemEval Task A Balanced 100 | Positive:100; Neutral:100; Negative:100 | Positive:3000; Neutral:3000; Negative:3000 |
| SemEval Task A Balanced 300 | Positive:300; Neutral:300; Negative:300 |
| SemEval Task A Balanced 1000 | Positive:1000; Neutral:1000; Negative:1000 |
| SemEval Task A Unbalanced 100 | Positive:111; Neutral:139; Negative:50 | Positive:11892; Neutral:13399; Negative:4347 |
| SemEval Task A Unbalanced 300 | Positive:345; Neutral:414; Negative:141 |
| SemEval Task A Unbalanced 1000 | Positive:1204; Neutral:1356; Negative:440 |
| SemEval Task CE Balanced 100 | Positive:100; Neutral:100; Negative:100 | Positive:1000; Neutral:1000; Negative:1000 |
| SemEval Task CE Balanced 300 | Positive:300; Neutral:300; Negative:300 |
| SemEval Task CE Balanced 1000 | Positive:1000; Neutral:1000; Negative:1000 |
| SemEval Task CE Unbalanced 100 | Positive:144; Neutral:126; Negative:30 | Positive:4014; Neutral:3576; Negative:1173 |
| SemEval Task CE Unbalanced 300 | Positive:404; Neutral:389; Negative:107 |
| SemEval Task CE Unbalanced 1000 | Positive:1335; Neutral:1317; Negative:348 |

We then take our classifiers, the classifiers we use are documented in table 2. For each classifier we train on every dataset in table 1. The exact setups of each model are chosen by fine to tuning on a validation set and referring to the literature on optimal configurations, we explore this a bit more in section 3 and give the final hyperparameters in section 4.

*Table 2*

|  |  |  |
| --- | --- | --- |
| Classical ML | Naive Bayes | SVM |
| Transfer Learning: Fine Tuning | AWD-LSTM ULMFiT | BERT-base |

4.1. The metrics

As our end goal is to simply compare classical machine learning to deep transfer learning we aggregate a lot of the metrics presented in section 5 by a simple arithmetic mean (full individual results are displayed in the appendix). We present 8 metrics per model per tier totalling 32 results. The exact metrics and how the are defined from the underlying datasets are given below.

**Clean Balanced**: Average of the three different clean trained unbalanced classifiers F1-micro scores as returned on the same dataset’s corresponding held out test set.

**Clean Unbalanced**: Average of the three different clean trained unbalanced classifiers F1-micro scores as returned on the same dataset’s corresponding held out test set.

**Clean Cross Clean**: Average of the three different clean trained balanced classifiers F1 scores as returned on the other two clean dataset’s corresponding held out test set.

**Clean Cross Twitter**: Average of the three different clean trained balanced classifiers F1 scores as returned on the two Twitter dataset held out test sets.

**Twitter Balanced**: Average of the two different twitter trained balanced classifiers F1-micro scores as returned on the same dataset’s corresponding held out test set.

**Twitter Unbalanced**: Average of the two different twitter trained unbalanced classifiers F1-micro scores as returned on the same dataset’s corresponding held out test set.

**Twitter Cross Clean**: Average of the two different twitter trained balanced classifiers F1 scores as returned on the three clean dataset’s corresponding held out test set.

**Twitter Cross Twitter**: Average of the two different Twitter trained balanced classifiers F1 scores as returned on the other Twitter dataset’s corresponding held out test set.

5. Experiments

We setup the models based on the referenced papers and fine tuning on the validation sets. For time constraints we only optimized on set of hyperparamters and pre-processing stages for each model, then all models were trained on this configuration. We looked at the amazon movies and Semeval task A: 100 and 1000 to optimize the hyperparameters before using these standard across all remaining datasets.

5.1. Naive Bayes

We mainly followed the approach laid out in [1]. Handling negation by appending a “not\_” to any word being negated and removing the original negation term. We also followed their suggestion of using a Bernoulli term frequency matrix and including bi-grams and tri-grams as all of these methods we independently found did boost the accuracy on the validation sets. We also experimented with number of features and found empirically that 30x the number of examples per class gave the best results. We present our gains on the validation sets in table 4, also included were the papers original gains on binary sentiment classification. We also gained a small increase by stemming but not by feature selection with Part of Speech (PoS) tagging, although perhaps with more manual feature selections gains could be made doing this. The final code can be found here [14].

Table 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Amazon Movies 100 | Amazon Movies 1000 | Semeval A 100 | Semeval A 1000 | IMDb[20] binary |
| Original |  |  |  |  | +23.77 |
| Negation |  |  |  |  | +9.03 |
| Bernoulli |  |  |  |  | +0.86 |
| N-grams |  |  |  |  | +1.54 |

5.2. SVM

[1]

5.3. ULMFiT

One of the immediate benefits of fine tuned transfer learning approaches is that complex features representations are transfered, this means that a lot of the effort in preprocessing and feature selecting is not required (or even possible) in these cases. Furthermore an additional benefit is that the models are designed to need minimal task specific architecture adjustments again reducing the amount of parameter selection needed on a validation set. In our experiments here we use the pre-trained AWS-LSTM weights published with the original paper, and continue with the proposed 2 dense layer append for the classification task, with a hidden state size of 50.

This leaves us with 4 choices of hyperparameters: the number of epochs for fine tuning and classification training, and the corresponding base learning rates. We present the results found on our 4 searches in the validation sets in table 5. WHICH HYPERPARAMETERS DID WE CHOOSE

Table 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Amazon Movies 100 | Amazon Movies 1000 | Semeval A 100 | Semeval A 1000 |
| Stage 2 epochs |  |  |  |  |
| Stage 2 learning rate |  |  |  |  |
| Stage 3 epochs |  |  |  |  |
| Stage 3 learning rate |  |  |  |  |

It should be noted that in stage two of training a ULMFiT we can train on more domain data than we have labelled, here we chose not to do this to give the worse case scenario for people using these models.

5.4. BERT

Again with transfer learning a benefit is that minimal task specific hyperparameters are needed and we do not need to select features. As such we go with the standard approach used in the original paper of attaching a single dense layer to the end of BERT. We are using the pretrained BERT-base model, in the original paper significantly better results we obtained with the larger model however we cannot run that do to hardware requirements, BERT-base already requires 12GB of VRAM and we are running on consumer grade hardware. All evidence suggests any results here would only be improved upon by users capable of running BERT-large. We use the uncased version.

Similar to ULMFiT we are left only choosing the learning rate and number of epochs for the classification phase, however in BERT only 1 phase is ran so we only need to select two hyperparameters.

BERT HYPERPARAMETERS

6. Results

We display the results for all of our experiments in table 5.

Table 5

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Clean Balanced | Clean Unbalanced | Clean Cross Clean | Clean Cross Twitter | Clean Summary | Twitter Balanced | Twitter Unblanaced | Twitter Cross Clean | Twitter Cross Twitter | Twitter Summary | Summary |
| Naive Bayes |  |  |  |  |  |  |  |  |  |  |  |
| SVM |  |  |  |  |  |  |  |  |  |  |  |
| Classical ML Summary |  |  |  |  |  |  |  |  |  |  |  |
| ASD-LSTM ULMFiT |  |  |  |  |  |  |  |  |  |  |  |
| BERT-base |  |  |  |  |  |  |  |  |  |  |  |
| Transfer Learning Summary |  |  |  |  |  |  |  |  |  |  |  |

Table 4a showing the results of the experiments with an average of 100 labelled examples per class

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Clean Balanced | Clean Unbalanced | Clean Cross Clean | Clean Cross Twitter | Clean Summary | Twitter Balanced | Twitter Unblanaced | Twitter Cross Clean | Twitter Cross Twitter | Twitter Summary | Summary |
| Naive Bayes |  |  |  |  |  |  |  |  |  |  |  |
| SVM |  |  |  |  |  |  |  |  |  |  |  |
| Classical ML Summary |  |  |  |  |  |  |  |  |  |  |  |
| ASD-LSTM ULMFiT |  |  |  |  |  |  |  |  |  |  |  |
| BERT-base |  |  |  |  |  |  |  |  |  |  |  |
| Transfer Learning Summary |  |  |  |  |  |  |  |  |  |  |  |

Table 4b showing the results of the experiments with an average of 300 labelled examples per class

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Clean Balanced | Clean Unbalanced | Clean Cross Clean | Clean Cross Twitter | Clean Summary | Twitter Balanced | Twitter Unblanaced | Twitter Cross Clean | Twitter Cross Twitter | Twitter Summary | Summary |
| Naive Bayes |  |  |  |  |  |  |  |  |  |  |  |
| SVM |  |  |  |  |  |  |  |  |  |  |  |
| Classical ML Summary |  |  |  |  |  |  |  |  |  |  |  |
| ASD-LSTM ULMFiT |  |  |  |  |  |  |  |  |  |  |  |
| BERT-base |  |  |  |  |  |  |  |  |  |  |  |
| Transfer Learning Summary |  |  |  |  |  |  |  |  |  |  |  |

Table 4c showing the results of the experiments with an average of 1000 labelled examples per class

7. Conclusion

Which results were better

8. Discussion

We have seen in section 6 that xxxxxxxxxxxxxxxxxxxxxxxxxx. However there are a couple of other key points to consider when making a decision.

Firstly the availability of the core language model xxxxxxxxxxxxxx

Production resource requirements are another major concern. For example the small BERT model used here required 11GB of VRAM to fine tune, this is high end consumer hardware at the time this paper was written. Other models such as BERT large or OpenAI GPT-2 [9] are much too big to fit on consumer hardware. This makes it out of reach for many researchers and un desirable for companies who are trying to minimize costs and are often reluctant to build pipelines that rely on expensive GPU compute instances. Contrary to this the classical ML models are small and can be trained and ran on any modern laptop.

Low shot classification summary

Could use BPE as per openai

8. Bibliography

[1] NB <https://arxiv.org/ftp/arxiv/papers/1305/1305.6143.pdf>

[2] ULM <https://arxiv.org/abs/1801.06146>

[3] BERT <https://arxiv.org/abs/1810.04805>

[4] <https://arxiv.org/abs/1301.3781>

[5] what is a language model

[6] deep learning better than classical for large datasets

[7] amazon movies <https://snap.stanford.edu/data/web-Movies.html>

[8] <http://jmcauley.ucsd.edu/data/amazon/>

[9] <https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf>

[10] sentiment classification harder than other classification tasks

[11] word piece embeddings <https://arxiv.org/abs/1609.08144>

[12] EMLo

[13] Semeval 2017 <http://alt.qcri.org/semeval2017/task4/data/uploads/semeval2017-task4.pdf>

[14] code repo

[15] imagenet transfer learning <https://arxiv.org/abs/1310.1531>

[16] naive bayes being good for text classification

[17] classical better than deep learning at low volumes <https://www.researchgate.net/publication/327746807_A_Comparison_of_Classical_Versus_Deep_Learning_Techniques_for_Abusive_Content_Detection_on_Social_Media_Sites/download>

[18] cross domain sentiemnt analysis <https://www.cse.ust.hk/~qyang/Docs/2010/p751-pan.pdf>

[19] AWD\_LSTM <https://arxiv.org/abs/1708.02182>

[20] IMDb <https://www.aclweb.org/anthology/W02-1011>

[21] <https://arxiv.org/abs/1703.04009>

[22] <https://ie.technion.ac.il/~roiri/papers/3841-16937-1-PB.pdf>

[23] <https://link.springer.com/chapter/10.1007/978-3-319-06483-3_25>