STUDIES ON DYNAMIC LOSS FUNCTIONS AND CURRICULUM LEARNING IN OFFENSEVAL DATASETS

Zhanfan Zhou

A thesis
in
The Department
of
Department of Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Science (Computer Science)
Concordia University
Montréal, Québec, Canada

September 2021

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Zhanfan Zhou**

Entitled: **Studies on Dynamic loss functions and Curriculum learning in OffensEval datasets**

and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

_____ Chair

Dr. Tse-Hsun Chen

_____ Chair

Dr. Eugene Belilovsky

_____ Supervisor

Dr. Sabine Bergler

Approved _____

Leila Kosseim
Chair of Department or Graduate Program Director

_____ 2021 _____ _____

Mourad Debbabi, Ph.D., Dean
Faculty of Engineering and Computer Science

Abstract

Studies on Dynamic loss functions and Curriculum learning in OffensEval datasets

Zhanfan Zhou

The spread of offensive language has become a severe social problem and may stress unmeasurable mental health illnesses. The rapid usage of social media worsens the situation. We develop a lite but robust offensive language identification system and evaluate the system on two SemEval offensive language identification shared tasks: SemEval 2019 Task 6 and SemEval 2020 Task 12. In order to take the advantage of a large semi-supervised dataset, and reduce the processing complexity of such huge data, we investigate approaches to adapt a model to the silver standards via curriculum learning and dynamic loss functions. By adapting a model to such data with the curriculum learning or dynamic loss functions, the systems are capable of scattering the focus properly on data of different difficulty levels. Experiments show both help the model learn effectively and acquire more messages from the hard cases without impairing the performance on easy cases. The best run on each task achieves competitive F1 score of 81.6% and 91.7% on the official test data of SemEval 2019 Task 6 and SemEval 2020 Task 12 respectively with at least 50% parameters and less data overhead, compared to the state-of-the-art systems.

Acknowledgments

This thesis was carried out with the selfless help of my supervisor, Dr. Sabine Bergler who guide me with her rich experience, knowledge, and distinct flair over the past 3 years. I am sincerely grateful to her for the endless supplies of patience and subtleties. I thoroughly enjoyed my time in the CLaC lab where I met so many individuals who were truly unique and wonderful in every way that mattered. To: Parsa, Narjes, Nihatha, Mingyou, Nadia S., Nadia B., Benjamin, and Claire, I will always appreciate all the things we learned and times we spent together. I would like to especially thank Parsa for giving many invaluable insights on my thesis, as well as sharing personal experiences to help me further.

I owe deep gratitude to my parents who supported me every way possible, and every step along the way; I love them beyond words. I also feel very fortunate to have tremendously supportive family members both back home and here in Canada, especially the latter who made my transition into a brand new environment that much smoother and my cousin Shiwei who offered suggestions on my writing.

The final thoughts go out for my best friends. I would like to thank Jiawen who brought my mood up with accompanying, and made me feel warm inside with the appealing memories we shared. I also need to give a shout-out to my awesome homie Gaoshuo and all the great friends with whom I shared countless joy and tears. My appreciation goes to Xu He for cheering me up on the gloomiest of days, making me feel at ease with the lovely moments.

To all: thanks for all the love during the journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Social media such as Twitter, Facebook and Reddit, unprecedentedly expose people to myriad content which may contain a considerable amount of verbal cyber-bullying such as offensive language, toxicity, hate speech, and abusive language. Anonymity on these platforms makes it more likely for users to post aggressive content [Festinger et al., 1963]. Manual efforts on verbal cyber-bullying identification tasks are unrealistic. Not to mention that the tremendous time would be devoted, long-term exposure to such content causes severe mental health aftermath to the annotators. This motivates studies on automatic detection of the verbal cyber-bullying [Xu et al., 2012, Gitari et al., 2015, Burnap and Williams, 2015, Malmasi and Zampieri, 2017, Zampieri et al., 2019a]. The most basic approach is gazetteer based models [Gitari et al., 2015] where trigger terms such as swear words and hate terms are precompiled as a filter. However, these gazetteer based approaches are insufficient for the detection due to the fact that the user generated content is steadily increasing and the context, social factors, events could largely influence our judgment on the offensiveness [Schmidt and Wiegand, 2017]. Moreover, the user-created new words and oscillating real world trending also make these systems vulnerable. Thus, leveraging deep learning methods in this field has increasingly drawn attention. Given multitude terminologies, some researches studied the differences and the overlaps between these concepts: hate speech, abusive language, offensive language etc. [Waseem et al., 2017, Davidson et al., 2017, Poletto et al., 2021]. Among these fields, prior studies have investigated such as hate speech: [Burnap and Williams, 2015, Davidson et al., 2017], Facebook posts: [Kumar et al., 2018] and offensive language:

[Zampieri et al., 2019a, Wiegand and Siegel, 2018]. However, the datasets they leveraged are rather small, and biased to certain social groups. This thesis focuses on the identification of offensive language, using a huge unbiased open-domain semi-supervised dataset.

1.1 Objectives and Motivation

When the data on social media is accessible within reach, the annotation for the numerous data is the main challenge. Apart from the post-traumatic effects on the annotators, it is intrinsically hard to obtain a large scale of high quality gold standard due to the multifold influences from linguistic and social phenomenon [Waseem, 2016, Laaksonen et al., 2020]. To side step the annotation issues, [Rosenthal et al., 2020a] addresses the limitations by creating a semi-supervised dataset for the offensive language detection task. However, the semi-supervised dataset (SOLID) is very large and provides silver standards. Since it was collated from Twitter without any filter options, it is comparatively unbiased but lacks corpus linguistic design and is highly imbalanced. It has limited offensive language patterns and together with the massive size it is unpracticable to use the whole dataset. Thus, it is nearly unusable without proper preprocessing such as down-sampling, folding, and resampling, etc [Zampieri et al., 2020]. The silver standard of SOLID is provided via four machine learning systems including a basic point-wise mutual information (PMI) model, neural network based models and the powerful pre-pretrained language model BERT [Devlin et al., 2019]. Deep learning has greatly promoted the field of NLP and the pre-trained language models such as BERT has achieved milestone progress in many natural language understanding tasks. The significant achievements give rights to automatic annotation, which largely facilitates the annotation efforts in terms of time and spend. Regarding the subjectivity of manually annotating, the automatic annotation even bypasses the deficiency. However, lack of investigation on the silver standard generated impedes the possible contributions from semi-supervised data. Given the corrupted annotations and the inadequacy mentioned, lots of work on offensive language identification discard the dataset or the noisy labels. Although these systems do not take the advantage of these labels or the whole dataset, they still yield promising results [Zampieri et al., 2020]. We owe this to the over simplicity

of the test data, with which a baseline system can achieve promising performance. And this also narrows the chances for improvement.

To confront the massive but questionable data, this thesis leverages curriculum learning and dynamic loss functions in our offensive language detection system. Curriculum learning has been successfully employed in noisy datasets [Jiang et al., 2018] and well-annotated datasets [Saxena et al., 2019] of image data. It simulates the human learning process and effectively enhances the learning efficiency. However, we find very few studies in the field of NLP, thereby we attempt to investigate the offensive language detection with curriculum learning. In this thesis, the dynamic loss functions and the curriculum learning share the same purpose, that is to adapt to the large amount of semi-supervised data. To the best of knowledge, most of the existing work investigates the supervised learning in this field but now more with semi-supervised data. It is true that using the semi-supervised data is not the best optimal approach in terms of targeting the classification task. But our goal is not to attempt the SOTA in the task. Thus, it could be the second best option in the sense of reducing the processing time and human annotation efforts as automatically as possible. Given the abundant accessible semi-supervised data, our goal is to build a classification system on the top of heterogeneous data sources: supervised and semi-supervised. By adapting the model to the semi-supervised data, the system requires less human annotation work and yields promising performances on data collected from 2019 and 2020.

1.2 Contributions

The main contributions are primarily listed as follows:

1. ways to use a rather large semi-supervised data with silver standards are explored;
2. the effects of curriculum learning and dynamic loss function on offensive language detection are investigated;
3. building a offensive language classification system by adapting the supervised and semi-supervised data;
4. competitive results on the shared tasks for offensive language identification via a light but robust model.

1.2.1 Thesis Outline

This thesis is orchestrated in the following order: the background knowledge is provided in Chapter 2 where the language features of Twitter and offensive posts are summarized. Moreover, the deep learning in NLP and the recent prevalent techniques are reviewed. This thesis emphasizes on reviewing some work on curriculum learning and dynamic loss functions and give detailed illustration on some of key methodologies. Afterward, the existing systems on offensive language identification and the datasets are introduced. In Chapter 3, the datasets are first compared and analyzed; then the experiments is reported; models on curriculum learning and dynamic loss functions are presented respectively. Chapter 4 concludes the study and discusses future work.

Chapter 2

Background

2.1 Tweet Language

As social media has been a massive part of our lives, people nowadays are overwhelmed with tons of messages. Meanwhile, we spread out our ideas, thoughts, feelings with simple taps on our phones. Twitter is one of such popular platforms where we post tweets for expression. Almost without any constraints, tweets cover diverse topics, formal or informal. It is possible to have a wide range of dictions, phrasing styles. Utterances may have spelling errors unintentionally or intentionally, grammar mistakes, abbreviations, acronyms, elongation, dog whistle terms, all-capitalization, slangs, emoticons, etc. To increase the communication diversity, Twitter provides various emojies. Users are allowed to insert URL links, create hashtag to make a topic trending, and mention other users by adding @ to an account. We also enjoy creating new phrases, some of which survive and become common phrases and some perish. To show some of the aforementioned language phenomena, some real examples are given to show some of the mentioned language phenomena.

Example 1. *@USER*[1] *@USER Go home you're drunk!!! @USER #MAGA #Trump2020* 👊 🇺🇸 👊 *URL*

It is normal to have redundant *@USER* mentions. The hashtags are consist of a sequence of unbroken words, which is sometime confusing at the first sight. *MAGA* is

[1] All user account mentions are normalized to @USER for anonymization.

an acronym for Make America Great Again.

Example 2. *@USER ajsjjsjdkkdjdk i want him so much he'd literally tell me to die and i'd obey him*

where *ajsjjsjdkkdjdk* is an intentional typo.

Example 3. *@USER she is idk what's going on #Qanon*

where *idk* is an acronym for I Don't Know. Such acronyms or abbreviations are prevalent on Twitter. Similarly, *#Qanon* or simply *#Q* refer to a far right conspiracy theory which readers without any background find it hard to understand.

Example 4. *@USER Varda is my fav valar. She is badass I bet melkor agrees with me also xd*

where *fav* is the abbreviation for Favourite and *xd* is a variant of the emoticon XD, which is signalling happiness or laughter.

Example 5. *@USER I love you bitchhh I aint never gonna stop loving you.... bitchhhhhh*

where *bitchhhhhh* is an elongation.

2.2 Offensive Language

This section presents the general definition of offensive language with demonstration of a few typical examples. The goal is not to provide a precise definition or distinguish it from related terms such as hate speech, abusive language, or aggressive language. Because of the intrinsic complexity in language, these terms mingle and overlap with each other very often. Although hierarchically they are in different level of specificity, the boundaries for these terms are relatively vague [Poletto et al., 2021]. This section emphasizes on revealing the general challenges on offensive language identification, knowing which may shed light on our studies, further experiments and analysis.

[Waseem, 2016] studied the influence of annotators' knowledge on hate speech classification task by training the system with data annotated by expert or amateur annotators. They found that the annotation agreement within amateurs are low and of high standard deviation. When they compared the annotation agreement of expert with an existing annotated dataset, they also saw low agreement. The disagreement

suggests that the hate speech dataset annotation is not an easy task and to obtain the high quality gold standards, extra efforts must be made. We believe this conclusion can be extended to the annotation of offensive language data as well.

[Reyes et al., 2012] investigated the figurative language of social media. Specifically, they studied two domains of figurative language: humor and irony by analyzing tweets. Figurative language in tweets implies meanings beyond the text per se. Different readers may infer the figurative language by their race, color, ethnicity, gender, cognitive level, etc. Thus it conveys distant semantics, sometimes offensive. Thereby, it sets great challenges to the detection of offensive language.

In addition, due to the nature of tweets, lack of context is also an issue. Furthermore, we have witnessed a lot of word sense changing chronologically. Many words used to be offensive but less nowadays and vice versa. We herein provide a few examples to show how the language complicates the offensiveness detection.

Example 1. *#Feinstein* ***you cant say anything truthful. You are a loers and liar****.*

Example 2. ***#Liberalismisamentaldisorder****, but it is also a* ***danger to our country****.* ***#Liberals are showing themselves to be utterly hateful****. They do not CARE about TRUTH, they only care about POWER.*

Two examples above are obvious offensive toward an individual or an organization and has strong signals of offending. The offensive parts are indicated by bold.

Example 3. *#FF @USER* ***he is an alien from the future*** 🙌🏻 👽 🌆 *URL*

Example 4. *#GreatestThingsAboutThe90s* ***liberals weren't unhinged***

Example 5. *@USER @USER Ppl who say I'm not racist are racist.* ***You Are A Racist****. Repeat after me*

Example 3 contains metaphors and Example 4 is being humorous or ironic. The last example is too vague to determine the offensiveness. The intriguing parts are indicated by bold.

2.3 Linguistic Notations

This section covers the linguistic notions that are fundamental in NLP and these notions are employed for dataset language comparison.

Tokenization Word tokenizer splits a sequence of text into single words, which is usually the foremost preparation for NLP. Lots of tokenizers have been developed for diverse text genres with different purposes. The prevailing pre-trained language model BERT [Devlin et al., 2019] is embedded with the WordPiece tokenizer [Wu et al., 2016]. Unlike the regular tokenizers that merely separate words, other than that, WordPiece may splits a word into subwords based on frequent sequence patterns. The example below shows how a sentence is tokenized by WordPiece. ANNIE [Cunningham et al., 2002] is another well-known domain independent tokenizer. For its stability and robustness, we use ANNIE for analyzing our dataset, which will be covered in the experiment section.

Part of Speech (POS) From the perspective of English grammar, a POS is a set of words sharing the similar grammar properties. In the field of NLP, POS tagging assigns each word token a POS tag from a tag set. Other than those typical English POS such as noun, verb, article, adjective, preposition, pronoun, adverb, conjunction, and interjection, the tag set contains finer tags with a lot more morphological subcategories. The most commonly used POS tag set is the Penn Treebank tag set [Marcus et al., 2002] which is primarily consist of 36 POS tags.

Typed Dependency The typed dependency provides binary relations among the words in an utterance. It sheds some light on representing syntactic structure of a sentence. The link of a dependency relation holds between a head word (governor) and a dependent (modifier). Below is a example of how typed dependency parses a sentence.

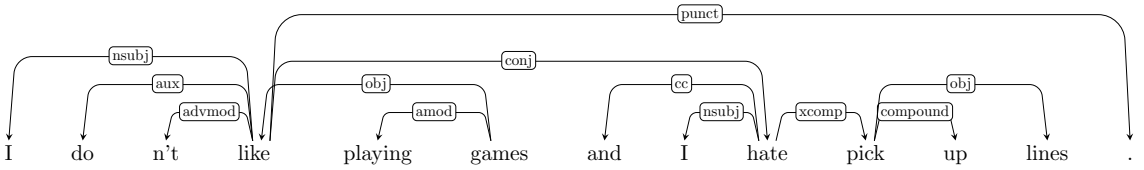

Figure 1: A Typed dependency example

Zips's Law Zips's law examines the frequency of words in natural language and

how the most common word occurs twice as often as the second most frequent word, three times as often as the subsequent word and so on until the least frequent word [Zipf, 2013].

2.4 Deep learning in NLP

The filed of NLP is being developed rapidly as deep learning has been dominating the field over the recent years. To illustrate the background of deep learning in NLP, we abstract a neural network as a function $f(\cdot)$. In general, the function takes in the word representations [2] as inputs and outputs task specific targets. Let $X \in \mathbb{R}^{T \times d}$ be the input word representations where T is the number of tokens in the input and d the size of word representations. $f(X)$ represents the basic form of a neural net and let $z = f(X), z \in \mathbb{R}^c$ be the output of the network where c is the number of classes. In common practices, the word representations are converted from a sequence of words which can be a sentence, or any text we feed in the neural network. To obtain the prediction of a neural net, a softmax function is often applied to the logits:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \tag{1}$$

The logits are denoted as z and z_i is the logit for the class of index i. For a classification task, the logits are transformed to the probability of each class. Next, we will illustrate some common neural network structures and word representation approaches.

2.4.1 Neural Networks

For years, researchers have been exploring powerful network structures for NLP tasks. The feed-forward network is one of the fundamental work of which form is linear $f(X) = XW + b$. The W is the weights and b is the bias term. Not taking into account the word order is a drawback of the feed-forward network. Taking the word order into consideration, the Recurrent Neural Network (RNN) [Elman, 1990], is composed from a sequence of recurrent units unfolded from a single unit through time. It produces sequential outputs and each recurrent unit is based on a small feed-forward net. Formally, a recurrent unit of the RNN is written as $h_t = f(x_t, h_{t-1})$,

[2] in form of vectors or matricies

where the t denote the index of the input sequence, x_t is the word representation at position t and h_t is the output of a recurrent unit, namely hidden states. In all, we abstract the RNN as $H = f(X, h_0)$, where $H = [h_1 \oplus \cdots \oplus h_T]$ is the row-level concatenation of all hidden states. When the sequence becomes too lengthy, it may raise the long-range dependency issue in which the dependency of two distant units exponentially decay and effect very little on each other eventually. To alleviate this, [Hochreiter and Schmidhuber, 1997] proposed long short term memory networks (LSTMs) where cell states are employed to cope with the long-range dependency issue. Sharing the sequential structure with the RNN, LSTMs can maintain information in memory for long periods of time with the cell states. In addition, to get better control of the inputs, outputs, and cell states, a set of gates are added to the LSTM blocks which allows the LSTM to decide what information to memorize, to output and to forget.

Convolutional Neural Networks (CNN) are one of the milestones and also were first used in the field of computer vision and later CNN was introduced in NLP successfully by [Kim, 2014] The convolutional blocks enable the CNN to capture the spacial information among words, and it yields promising results in NLP tasks.

Transformers, proposed by [Vaswani et al., 2017] launched a new era of NLP. It is based on sequence-to-sequence architecture which transforms a sequence of input into another sequence. The Transformer consists of a stack of encoders and decoders. Figure 2 shows the structure of a Transformer encoder.

Transformer architecture ditched the recurrent structure in favor of multi-head self-attention mechanism. The input word representations $X = [x_1 \oplus \cdots \oplus x_T]$ is a row level concatenation of input word representations, denoted as $x_t (t = 1, \ldots, T)$. To embed the positional information, in addition to the input word representations, positional encoding is added to the input. After obtaining the positional encoding, the inputs are fed into the Multi-head self attention block which is based on scaled dot-product attention. To explain the Multi-head self attention mechanism, we start from scaled dot-product attention. Given queries Q, keys K, and values V of dimension d_k, the scaled dot-product attention is given by:

$$\text{scaled dot-product attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2}$$

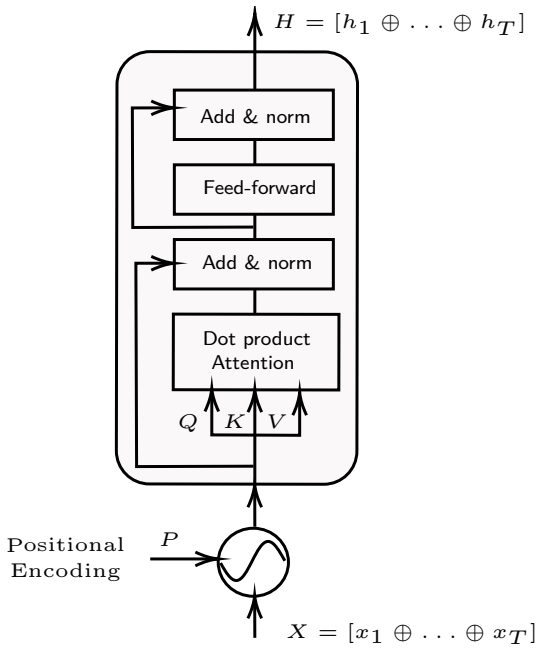

Figure 2: Transformer Encoder

where K^T is the transposed K and $\sqrt{d_k}$ is a scaling factor which prevents the dot products of Q and K^T growing large in magnitude then overwhelming the softmax function. The intuition behind scaled dot product attention is that the first part softmax($\frac{QK^T}{\sqrt{d_k}}$) produces an element-wise similarity matrix scaled by $\sqrt{d_k}$. The similarities decide how the output is composed by V. Under the self attention setting, Q, K, V are computed by the same input. That is: $Q = xW_q$; $K = xW_k$; and $V = xW_v$ where x is the input word representations and Ws are learnable weights. To perform the Multi-head, one can split the W_q, W_k, W_v into n parts evenly and perform the scaled dot-product attention to the input n times. The output of each attention head is then concatenated and applies a linear transform. The formal expression is given by:

$$\begin{aligned} \text{head}_i &= \text{scaled dot-product attention}(Q_i, K_i, V_i) \\ \text{Multi-head self attention}(Q, K, V) &= \text{Concat}(\text{head}_1, ..., \text{head}_n)W_0 \end{aligned} \tag{3}$$

where W_0 is the added linear transform and Q_i, K_i, V_i are obtained via the split W_q, W_k, W_v respectively. Afterward, the residual connection adds the input word representations to the attention output and performs a normalization operation. Note that to fit the shape of a sequence, the feed-forward net is positional-wise. The output of a Transformer encoder block is defined as hidden states H which share the same

shape of input word representations. Each word of the input x has its corresponding encoded output h. Thus, the Transformers can be stacked on top of each other easily.

The self attention mechanism enables the Transformer to capture contextual information bidirectionally or better non-directionally. Unlike the recurrent architectures, which can either receive the input from the previous or reversely from the future, Transformers gather the contextual semantics for each input word. With the relatively simple architecture (feed-forward neural networks and attention mechanisms), the Transformer usually yields better results than recurrent models like RNNs and LSTMs.

2.4.2 Word Representations

As the input of a neural net, lots of approaches exist for generating word representations.

Early studies represent words using the One-hot representation, in which the input words are often encoded as a 0/1 vector. Each entry in the vector is zero except the word co-occur in the input are set to ones. To this end, one should first create a word inventory to store the vocabulary of all the inputs and the size of the word representation vector is the size of the vocabulary V, $X \in \{0,1\}^{|V|}$. The One-hot representation catalyzes the Bag-of-Words (BoW) model [Harris, 1954] which describes the occurrence of words in a count vector. Although it is straightforward and convenient to use, the One-hot representation loses the word order and is usually a sparse vector in which the zero values account for most of the vector entries. Because the vocabulary size is rather large but the input text is relatively short and consists of a small amount of vocabulary.

Other than the One-hot representation, co-occurrence matrices represent a word by the words co-occurring in a same context. The matrix is symmetric and is of shape $|V| \times |V|$ in which each entry represents a word and each cell is the number of times the row word co-occurs with the column word. Here V is a set, and $|V|$ represents the cardinality of the set. One deficiency is the excessive matrix size. Having the co-occurrence matrices treat every word equally, TF-IDF is an approach to build document representations based on the word importance. The document representations is composed of the representation of words that form the document. Each word that occurs in the document has its term frequency (TF) and inverse document frequency (IDF) score. The product of TF and IDF of a word is used as

a weight to represent the word. The TF-IDF score implies the importance of the word in the text and corpus. The higher score, the more important the word could be. However, although TF-IDF is more informative than One-hot representation, it eventually generates a sparse vector for the inputs.

It is problematic to use the sparse word representation vectors, since the large size vector impedes the computation efficiency and lack of word order, word sense information, and semantics. Addressing these issues, dense word representations, often called word embeddings are proposed.

2.4.3 From Word Embeddings to Pre-trained Language Models

Word embeddings have proven to be powerful in many text classification tasks [Sun et al., 2020, Liu et al., 2019a, Wiedemann et al., 2019]. The earlier representative efforts on word embeddings such as Word2Vec [Mikolov et al., 2013], GloVe [Pennington et al., 2014] are pre-trained word embeddings, aiming at converting words to context independent vector space representations. They are trained on large scale text corpora by an unsupervised approach and are often used as the input to a neural net. Such context independent word embeddings are available to be used off-the-shelf and map each word into a vector representation regardless of the context. In order to embed contextual information in the embeddings, deep contextualized word representations (ELMo) is proposed by [Peters et al., 2018]. In contrast to the previous word embedding models such as Word2Vec and GloVe, these context-sensitive word embeddings are able to capture the sequential information of the input words and dynamically provide context dependent word representations. Note that ELMo takes in character-level inputs and therefore to some extent embeds morphological information. Different form the previous word embeddings, ELMo is derived from a bidirectional LSTM training with the language model (LM) objective on a large scale corpus. Briefly, a traditional LM task learns to predict the next word in a sentence based on the previously observed words. The pre-trained ELMo is often used for fine tuning [Dauphin et al., 2012] in down-stream tasks. The most common way to adapt a pre-trained language model is to fine-tuning it on task specific datasets. Pre-training and fine-tuning have been prevalent in recent years and have demonstrated to be superior to retrain a word embedding or a LM from scratch in lots of studies, and

some of them will be covered in next section.

[Devlin et al., 2019] proposed a contextualized language model **BERT**: Bidirectional Encoder Representations from Transformers. It has obtained new state-of-the-art results on eleven NLP benchmarks. BERT is derived from Transformers and is trained ob an unsupervised approach for masked language model (**MLM**) and next sentence prediction (**NSP**) tasks.

In contrast to ELMo, a uni-directional language model, BERT encodes each word of the input sentences based on the bidirectional surrounding context. Although EMLo leverages bidirectional LSTM structure, it still has the directional properties due to the fact that LSTM encodes the input sentence sequentially, either from the beginning to the end or in the reverse direction in one go. The Transformer base, however, allows BERT to capture the surrounding context in both directions (bidirectional). BERT is a stack of Transformer encoders. Thereby, it inherits the property of the Transformer.

BERT is trained with two objectives:

- MLM, where 15% of the tokens of the input are randomly masked and the model learns to predict only those masked tokens
- NSP, where a pair of sentences are given to the model who learns to predict whether the second sentence is the actual next sentence of the first one.

To perform the two tasks, BERT adds a special *[CLS]* token at the beginning of every sentence. The final hidden state of the Transformer corresponding to this token is taken as a sequence embedding vector for classification tasks. Since BERT has been already trained on the BooksCorpus (800M words) [Zhu et al., 2015] and English Wikipedia (2,500M words), one can take the advantage of the well-trained BERT models. For different purposes, [Devlin et al., 2019] released four BERT models differentiated by word case-foldings and parameter size, namely BERT-base-uncased, BERT-base-cased, BERT-large-uncased and BERT-large-cased. A brief summary of these models is provided in Table 1. To employ BERT on down-stream tasks, [Sun et al., 2020] found that further training of the BERT model with MLM helps the model adapt to the down-stream task. As [Devlin et al., 2019] suggested, fine tuning BERT by adding a classification layer on the top of BERT and using the final hidden state of *[CLS]* token as the input of the classification layer is also practicable.

Thanks to the success of BERT, a series of BERT like pre-trained language models are proposed. [Liu et al., 2019b] proposed a robustly optimized BERT pre-training

model (**RoBERTa**). Retaining the BERT-large architecture, RoBERTa optimizes BERT's training procedure and outperforms BERT on most of the benchmarks. Firstly, it is no longer trained with the NSP objective. Second, they increase the training data to 10 times as BERT's and train the RoBERTa with larger batch size and longer sequences. Third, instead of using the static mask for MLM training, RoBERTa dynamically changes the mask every time a sequence is fed in.

Despite the prominent achievement of BERT, operating such a heavy model is costly in inference and imposes huge stress on computational hardwares. Addressing these issues, [Sanh et al., 2019] proposed **DistilBERT**, a lower cost, faster, lighter pre-trained model. DistilBERT is derived from BERT using knowledge distillation. Retaining 97% of BERT's language understanding capabilities, the size of DistilBERT is reduced to 40% of BERT's. They take DistilBERT as a student model who learns to mimic the behaviour of the teacher model, in their case, BERT. The teacher-student training diagram transfers the knowledge from the teacher to the student via soft targets learning rather than training with hard targets (one-hot encoding of the true class). Thus, one of the key components of the loss function of DistilBERT is distillation loss, given in Eq 4, hoping the student model evolving the same way as teacher by mimicking the teacher's output distribution. It certainly enriches the learning materials compared with one-hot encoding target learning.

$$L_{\text{distillation}} = -\sum_i t_i \log(s_i) \tag{4}$$

where t_i and s_i are softmax-temperature for teacher model and student model which is given by:

$$\text{softmax-temperature}(z_i) = \frac{\exp(\frac{z_i}{T})}{\sum_j \exp(\frac{z_j}{T})} \tag{5}$$

where z_i is the model logit for class i and T is the temperature, controlling the smoothness of output distribution. As $T \to 0$, the softmax-temperature becomes more skewed and is equivalent to the one-hot target vector; Contrarily, as $T \to +\infty$, the function is flatten and the output is close to uniform distribution. If $T = 1$, it reduces to the standard softmax function. softmax-temperature is introduced by [Hinton et al., 2015] as a modification of softmax. The purpose is to explore more out of the granularities from the teacher model output distribution. In other words, it describes the sensitivity that the student model is influenced by the teacher.

Hinton released the DistilBERT model based on distillation from BERT-base. Table 1 compares the mentioned models by their model size.

Model	Transformer block	Parameter
BERT-base-cased BERT-base-uncased	12	≈110 million
BERT-large-cased BERT-large-uncased RoBERTa-large	24	≈340 million
DistilBERT	6	≈66 million

Table 1: Summary of released BERT pre-trained models

One limitation of these pre-trained LM is that they are trained for generic purpose tasks with open domain corpus. These LMs may not adapt ideally to the task specific language varieties. Therefore, re-training a BERT-like model has shown to be an effective and competitive approach in adapt the LM to a new domain [Barbieri et al., 2020]. This approach is practicable when the training data is not scarce to train a new BERT. HateBERT [3] is re-trained on the Reddit Abusive Language English dataset (RAL-E) following BERT training convention [Caselli et al., 2021]. The dataset contains Reddit Comments from December 2005 to March 2017, resulting in 1,492,740 messages after conducting data cleaning.

2.5 Underlying Concepts

2.5.1 Curriculum Learning

Curriculum learning (CL) was first proposed by [Bengio et al., 2009] and several applications have been published in the field of NLP and computer vision [Soviany et al., 2021, Wang et al., 2021]. Drawing inspiration from the cognitive learning process of human beings and animals, CL aims at presenting learning systems with appropriate learning materials at current stage rather than feeding randomly. Presenting learning materials to machine learning systems in a well organized order, CL achieves considerable progress such as enhancing learning quality, improving

[3]There exists more re-training BERT models in other domains such as HurtBERT, BERTweet etc. This thesis primarily introduces HateBERT.

robustness and better generalization. Thus, it has drawn increasing attention from researchers in recent years.

Common teaching practices reveal that learning always starts with the fundamentals, then progressively moves onto the more advanced field. With that plain and straightforward cognition, it is enough for junior students to solve many simple problems. When it comes to advanced, it requires involving sophisticated knowledge from multiple angles. Solving complicated puzzles need senior students who equip with skillful mindset and techniques. This learning paradigm is commonly applied in many CL studies. [Elman, 1993] found that the RNN learns synthetic grammar effectively when presenting the data from simple to hard order. The paradigm has been demonstrated to have better generalization by [Bengio et al., 2009, Kumar et al., 2010].

One of the main issues in applying CL to tasks is the definition of easy and hard cases. Most earlier studies carefully designed a ranking function that prioritizes the easy cases before the hard one following the easy-to-hard paradigm. The ranking function is usually manually defined by heuristic knowledge such as [Sachan and Xing, 2016]'s work or as simple as short-sentence-first principle [Kocmi and Bojar, 2017].

One of the curriculum that [Bengio et al., 2009] proposed is referred as one pass curriculum by [Cirik et al., 2016]. All samples are sorted into different difficulty levels of bins. Training starts from the simplest bin and it will not move to the next bin until the loss on a held out development set stops reducing for a certain epoch. The entire training process will be carried on until all the bins are trained.

The underlying problem for predetermined heuristics is that they are only reasonable to humans but may not be the case to machine learning systems. With this notion, [Xu et al., 2020] employed a difficulty evaluation module to estimate difficulty scores, instead of explicitly setting heuristic rules. Still, given that we are able to feed the learning system with appropriate examples, it is important to feed them in at the right time. For example, easy cases should garner less attention when the learning system had become mature. Thereby, [Saxena et al., 2019] proposed a dynamic curriculum learning manner with class parameters, which shows improvement on image classification and object detection. Additionally, the model outperforms the SOTA system in presence of noisy labels. In the field of neural machine translation (NMT), [Wan et al., 2020] employed a flexible manner to address the problem in which the NMT model is able to measure the confidence level over training examples and

therefore regulate the loss during training.

The above researches aiming to alleviate the drawbacks of CL, share the same purpose with self-paced learning (SPL). Proposed by [Kumar et al., 2010], SPL aims at alleviating

1. the mismatch of pre-defined fixed ranking function and the flowing learning system;

2. the lack of computational definition of difficulty of a training sample, by merging the learning curriculum into loss functions.

The loss function is dynamically changed along with the model.

To elaborate, [Kumar et al., 2010] defined the structure of learning objective for SPL. To approaching their end, a binary vector parameters v is added to standard loss function $l(\hat{y}, y)$, computing the loss between predicted label $\hat{y}$ and ground true label y. The parametrized learning objective is given by:

$$\min_{w \in \mathbb{R}^d, v \in \{0,1\}^n} \mathbb{L}(w, v; k) = \sum_i^n v_i l(f_w(x_i), y) + r(w) - \frac{1}{k} \sum_i^n v_i \tag{6}$$

where $f_w(\cdot)$ represents a model with model parameter w, it can be a neural network. $f_w(x_i)$ is model's prediction of sample x_i, and $r(\cdot)$ is a regularization term which penalize w for obtaining excessively large value. v_i is a binary variable, either 0 or 1. k is a weighting term that determines the current number of valid examples. With $-\frac{1}{k}\sum_i^n v_i$, the model is encouraged to include as many sample as they can, however, it may have a great loss value given brought by $l(f_w(x_i), y)$. Given a large k, $\frac{1}{k}$ down-weights $\sum_i^n v_i$ making it less important numerically whereas $v_i l(f_w(x_i), y)$ dominating the loss. To minimize $\mathbb{L}$, the loss function focuses on a very few the easy cases who tend to have small loss given by $l(f_w(x_i), y)$, minimizing $v_i l(f_w(x_i), y)$. As the k decreases, v_i is allowed to increase the number of ones, which is equivalent to include more training samples.

During the training, the k is decreasing gradually. They found that optimizing v using alternative convex search (ACS) yields better accuracy, that is:

$$v_i = \begin{cases} 1 & l(f_w(x_i), y) < \frac{1}{k} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

In general, [Kumar et al., 2010] embeds selective curriculum into the form of loss function, in which the parameters controlling the curriculum change with the model. Again, as suggested beforehand, it is significant to have non-monotonous learning, that is, the difficulty of sample varies across the course of training. It is the loss that determine the degree of difficulty. However, the binary nature of v and the predetermined k limit the training flexibility. As shown in Eq. 7, if the given loss is above the threshold, the corresponding sample will be discarded. It will not be included until the model is at the stage when it can produce a small loss w.r.t that sample. v_i acts as a gate mechanism filtering out the hard cases for current stage.

What [Saxena et al., 2019] differs from [Kumar et al., 2010] is that the loss functions is more flexible and dynamic. Specifically, samples can partially contribute to loss depending on the learning stage. Besides, it is possible to have different loss for different classes compared with [Kumar et al., 2010] in which the loss is fixed in $l(\hat{y}, y)$.

For the rest of this section, we will review the dynamic curriculum learning manner, which is proposed by [Saxena et al., 2019] in detail, including explaining the functionality behind, analyzing the drawbacks and how we employ the manner in our scenario.

For conciseness, we cover the technique that is relevant to the thesis. Each class has an additional parameter namely class parameters which describe the importance during training. Intuitively, class parameters assign a dynamic weight for each class indicating the importance at current learning stage. During back propagation, the class parameters are updated along with model parameters. This process leads to dynamic changes of class parameters as well as the loss function, which reflects the learning curriculum in favour of the learning system.

To elaborate, we define our dataset as $D = \{(x_i, y_i)\}, i \in \mathbb{N}$, where x is training samples and y denotes target classes. For a classification task of k classes, we have $y_i \in \{0, 1, ..., K\}$ and class parameters $\sigma_y \in \mathbb{R}^K$, where the each entry is the learnable weight of that class. Let $z_i \in \mathbb{R}^K$ be the logits predicted by the network and $L(\cdot)$ be the loss function. For clear explanation, we assume k the index of true label and z_i^k the logit for the ground true class. The class parameters are used to scale the logits

before the softmax happens:

$$softmax(z_i^k)' = \frac{exp(\frac{z_i^k}{\sigma_{y_k}})}{\sum_{j \leq K} exp(\frac{z_i^j}{\sigma_{y_j}})} \tag{8}$$

where $\frac{z_i^k}{\sigma_{y_k}}$ is the scaled logit for class k. The scaled softmax values $softmax(z_i^k)'$ are fed into the loss function $L(\cdot)$. Supposed the cross entropy loss is used, the loss for x_i is given by:

$$L(x_i) = -\log(softmax(z_i^k)') \tag{9}$$

The derivative of loss w.r.t σ_y is calculated as:

$$\frac{\partial L_i}{\partial \sigma_{y_k}} = \frac{1 - softmax(z_i)'}{{\sigma_{y_k}}^2}(z_i^k - \sum_{j \leq K, j \neq k} \frac{softmax(z_i^j)'}{1 - softmax(z_i^k)'} z_i^j) \tag{10}$$

and let:

$$\delta = z_i^k - \sum_{j \leq K, j \neq k} \frac{softmax(z_i^j)'}{1 - softmax(z_i^k)'} z_i^j \tag{11}$$

where the term $\delta = z_i^k - \sum_{j \leq K, j \neq k} \frac{softmax(z_i^j)'}{1 - softmax(z_i^k)'} z_i^j$ becomes negative if the logit of real label less than the sum of logits of other labels, which decides the direction of the gradient. In [Saxena et al., 2019]'s work, they update the class parameters with the negative value of the gradient using stochastic gradient descent (SGD). In such way, if the system correctly predicts a sample with $\delta > 0$, the gradient w.r.t that class parameter is a positive value. Thereby the class parameter σ_{y_k} is decreased when updating it with the negative value of the gradient, resulting in the up weighting of $\frac{z_i^k}{\sigma_{y_k}}$. It acts oppositely when $\delta < 0$.

Updating the negative gradient of loss w.r.t parameters is commonly applied to minimize the loss. At the beginning stage of training, the learning system prefers easy cases by up-weighting the class it has done well already. In their implementation, a regularization term as used in [Kumar et al., 2010] is added to loss L in order to prevent the system from focusing solely on easy classes. Therefore, as the training moves forward, σ_y of hard classes are decaying gradually with increasing of σ_y of easy classes. Because of the regularization term, the weights for easy class does not rise greedily. On the contrary, it strikes the balance between acquiring easy knowledge with large regularization penalty and turning the attention to learning hard cases with

low penalty. Normally, it requires a huge amount of training time and data before the switching. It takes 20 epochs in their experiments.

They mention that updating the class parameters using the exact value of the gradient will accelerate the learning, but they do not provide any relevant experiment. During the inferencing, the class parameters are detached from the model because the labels are inaccessible. The class parameters effect on the learning process and lay no burden on the model parameters and inference.

2.5.2 Continual Learning

Lots of existing work on machine learning trains the learning system in isolation. The resulting system may fail to achieve a reliable performance on different sources of data [Widmer and Kubat, 1994]. The fact is, we are continually exposed to innumerable types of data which could be varying in word distribution, annotation standard, and even word sense. It is neither time-efficient nor sustainable to retrain a model with new data. Thereby the fact of increasingly growing data requires our learning system having the ability to learn new knowledge based on what has learnt and avoids forgetting the previous.

Continual learning aims to confer a learning system the ability to incrementally acquire knowledge from a continuous data stream, hoping to leverage the previously learnt task to improve the current task. The main challenge is the **catastrophic forgetting** problem [McCloskey and Cohen, 1989]. Learning systems suffer from forgetting the well-learnt representations when trying to acquire knowledge from new tasks. It is also known as stability-plasticity dilemma (Abraham and Robins, 2005) in which models balance the trade-off between retaining previous knowledge and adapting to the new field [Hong et al., 2018, Biesialska et al., 2020]. Another relevant key issue is referred as capacity saturation [Sodhani et al., 2019] which emphasizes that the representation ability of a parameterized model is limited by a fixed maximum amount, where no more new representation could be embedded in net.

As illustrated in [Li and Hoiem, 2016] **Fine tuning** [Dauphin et al., 2012] and Multi-task learning are two of the prevailing manners to transfer knowledge from previous data. For fine tuning, it is common to have a slightly low learning rate when fine tuning on a new task, which prevents the weights from drastically shifting. Multi-task learning requires that all tasks are trained jointly and optimized the model

across tasks.

Although continual learning has been widely exploited in computer vision, to the best of our knowledge, the work remains nascent in the field of NLP [Biesialska et al., 2020]. [Kemker et al., 2017, Shin et al., 2017] leveraged the data distribution of previous tasks to generative models. [Greco et al., 2019] demonstrated that the catastrophic forgetting problem could be alleviated to a certain degree by reorganizing the order of task difficulty. The notion accords with the view of curriculum learning. Specifically, in [Greco et al., 2019]'s visual question answering tasks, wh-questions are believed to be easier than yes-or-no questions, since the former are simple and naive with fixed alternative answers space, whereas the later requires reasoning. It corresponds with the performance when separately train and test on these two tasks respectively. The test accuracy for yes-or-no questions are merely above random guess while the wh-questions are around 0.8. They found that precedding from wh-question (easy task) to yes-or-no questions (hard task) facilitates the course of training and yields better performance.

2.5.3 Loss Functions

Before getting into the functionality of loss functions, we start with introducing the role of loss function playing in deep learning. A loss function is also referred to as error function or cost function. What we say objective function is something more general [Goodfellow et al., 2016]. For a deep learning model, the loss function measures how well the model fits the data. In general, a model with lower loss fits the data better than the higher. Given a set of data points X and labels Y, we define the dataset as $D = \{(x_i, y_i)\}, i \in \mathbb{N}$. The goal of training the model is to minimize the loss over D. Formally, it is written as:

$$L(D) = \sum_i L(\hat{y}_i, y_i) \tag{12}$$

where $L(D)$ is the overall loss of D and $\hat{y}_i$ is the label predicted by the model. Supposed we use $f(\cdot)$ to denote our model and θ the model parameters to learn. The predicted label of data point x_i, $\hat{y}_i$ is given by $\hat{y}_i = f_\theta(x_i)$. In order to get the optimal parameter θ that minimizes the overall loss $L(D)$, we update θ with a small step to the direction of minimizing L at each iteration. To this end, we take the gradient of loss w.r.t θ ($\frac{\partial L}{\partial \theta}$) and renew θ with the negative value of the gradient. Hopefully, it may converge and

approximate the global minimum after many iterations. Supposed we use stochastic gradient descent (SGD) to optimize θ, θ_{t+1} is derived from θ_t following:

$$\theta_{t+1} = \theta_t - \eta \frac{\partial L_i}{\partial \theta_t} \tag{13}$$

where η is the learning rate, a small scalar controlling the learning step size and $\frac{\partial L_i}{\partial \theta_t}$ is the the gradient of loss given by sample x_i.

Cross entropy loss (CE) has been the most common used loss function among many machine learning studies. It claims to work best and become the first option for many systems. However, it has been proved not robust to noisy labels [Ghosh et al., 2017]. Further, [Wang et al., 2019] found that learning with CE may suffer from class-bias, in which some easy classes converge much faster than the hard classes causing the notable under-fitting of hard cases. This diverges from what is commonly believed that the underperforming is due to overfitting to noise.

To be specific, [Wang et al., 2019] investigated the accuracy of the prediction of intrinsically hard classes with and without noise injection on distinct learning stages (early, middle, and later). The results showed that with noise injection, test accuracy varied significantly across classes, whereas even without noise injection, the prediction confidence of hard cases is at low level across all stages. they concluded that CE by itself is not sufficient for hard case learning, noisy data will only worsen this situation. To address the issue, they proposed **Symmetric CE loss** (SL) which combines Reverse CE loss (RCE) and CE loss with weight parameters α and β:

$$l_{sl} = \alpha l_{rce} + \beta l_{ce} \tag{14}$$

The RCE is given by:

$$l_{rce} = -\sum_k p(k \mid x) \log q(k \mid x) \tag{15}$$

where $p(k \mid x)$ is the predicted distribution over label k, and $q(k \mid x)$ ground truth distribution of label k which is a constant in practice. We explain how SL alleviates that issue from the perspective of gradient. Let z_j denote the logits and k the index of true label. For clarity, we assume binary labels and $k = 1$. The derivative w.r.t to

z_j is give by:

$$\frac{\partial l_{sl}}{\partial z_j} = \begin{cases} \frac{\partial l_{ce}}{\partial z_j} - \epsilon(p_j^2 - p_j) & k = j \\ \frac{\partial l_{ce}}{\partial z_j} - \epsilon p_j p_k & k \neq j \end{cases} \tag{16}$$

where ϵ is a negative constant replacement for $\log 0$ and $p_j \in [0, 1]$. The core of SL is a scaled CE loss where $\epsilon(p_j^2 - p_j)$ and $\epsilon p_j p_k$ are the modulating factors. The former term increases the learning volume stepping toward true class, whereas the later term deducts the gradient from false classes.

Further, The same team proposed a normalization framework for normalizing any loss functions [Ma et al., 2020] addressing the similar issue. Another successful demonstration of scaling CE loss is Focal loss.

Focal Loss (FL) which is proposed by [Lin et al., 2017] and is widely used in object detection. The aim of FL is to address the class imbalance issue. They found that easily classified cases compose the majority of the loss and dominate the gradient. In object detection, it is normal to have a relatively large background and small objects that merely account for a small part of an image. Accordingly, Focal loss up-weight the loss of miss-classified samples by down-weighting the well-classified samples. Such that the system focuses more on hard, miss-classified samples. It has demonstrated success in dealing with the large class imbalance issue.

FL defines p_t to measure difficulty.

$$p_t = \begin{cases} p & \text{y=1} \\ 1 - p & \text{otherwise} \end{cases} \tag{17}$$

where $y \in \{0, 1\}$ denotes gold standard label. p is the model estimated probability for positive class. Then p_t can rewrite binary cross entropy loss (BCE):

$$BCE = \begin{cases} -\log(p) & \text{y=1} \\ -\log(1 - p) & \text{otherwise} \end{cases} = -\log(p_t) \tag{18}$$

Focal loss reshapes the binary cross entropy loss by adding a modulating factor $(1-p_t)^\gamma$ with a tunable parameter $\gamma \geq 0$, defined as:

$$FL(p_t) = -(1 - p_t)^\gamma \log(p_t) \tag{19}$$

The curves of FL with different *gamma*s is shown in Figure 3. We see that the outer curve is the standard CE loss where easily classified examples with large $p_t > 0.5$ can incur a loss with non-trivial magnitude. By adding the modulating factor, the standard loss is reduced for that easily well-classified samples, addressing more focus on those hard. Over all, FL reduces the loss contribution from the easy negatives and increases the hard positives. It measures the difficulty via the predicted logits with which it scales the BCE loss.

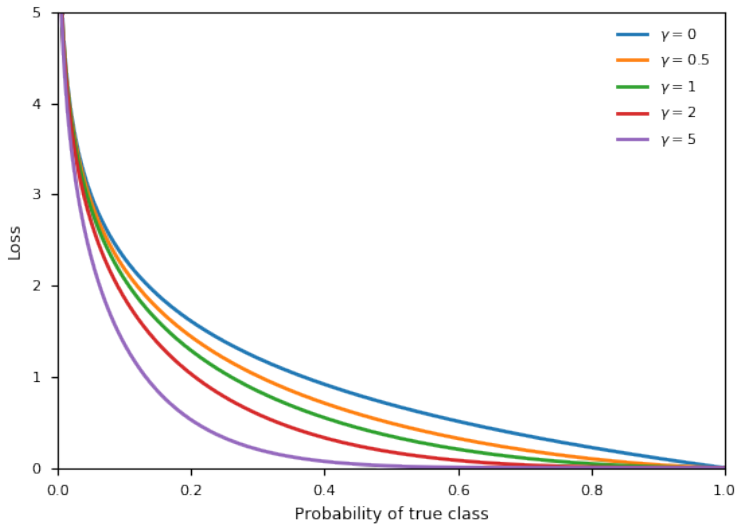

Figure 3: Plotting FL with different γ

Scaled loss functions enable the system to assign different weights to samples of different level of difficulties. Intuitively, hard samples should be given more weight, while the easy ones should be less considered since they are already well done. However, fixing or carefully designed may work for some scenarios, but not for all. Samples should not always be up-weighted or down-weighted. An elegant way is letting the system decide what sort of sample is the most informative and turn its focus onto it with **dynamic loss functions**.

Common teaching practices reveal that learning always starts with the fundamentals, then progressively moves onto the more advanced field. With that plain and straightforward cognition, it is enough for junior students to solve many simple problems. When it comes to advanced, it requires sophisticated knowledge from multiple angles. Solving complicated puzzles needs senior students who are equipped

with a skillful mindset and techniques. Fundamental knowledge is less-likely to get involved in directly when tackling higher level problems. However, it is necessary to learn advanced knowledge based on the fundamentals. During that process, it is also important to review the background knowledge before learning something new. Otherwise the senior student may suffer from forgetting the well-learned, which impedes them from moving forward.

[Wu et al., 2018] proposed a gradient based algorithm that optimizes the student model with dynamic loss functions outputted by the teacher model. The motivation of their work is the analogy between loss functions in machine learning and teachers in educating human students. Loss functions reflect the teaching strategies. Conducting exams during teaching reveals the learning quality of the student model as well as the effectiveness of teaching skills. Therefore, the student is guided by dynamic learning strategies, and the teacher keeps refining the teaching skills. Specifically in their work, student model is evaluated under a stand-alone development set with the loss function outputted by the teacher model. The gradient is computed then back propagated to optimize the teacher model using Reverse-Mode Differentiation (RMD).

For detailed illustration, we define student model with trainable parameter w as f_w, teacher model with trainable parameter θ as g_θ, L as the loss functions outputted by $g_\theta(\cdot)$. We view the training of the student model as a sequential process with T steps, in which w is gradually trained from initial values w_0 to w_T. Within a time step, the loss function is determined by $g_\theta(s_t)$ where s_t is a state vector of the student model which can be any feature representing the current training phase. For each training data point at time t, denoted as D_t, the derivative of the outputted loss $L_{g_\theta(s_t)}$ w.r.t w_t can be written as $\sum_{x,y \in D_t} \frac{\partial L_{g_\theta(s_t)}(f_{w_t}(x),y)}{\partial w_t}$, which is used to update w_t by any optimization algorithm. Similarly, we define $\tilde{M}(\cdot)$ as the evaluation function which acts like a loss function, to evaluate the student model at time step t.

When it comes to training the teacher model, RMD will loop backward from the last time step T. The gradient of w_T computed on development set can be written as $\sum_{x,y \in D_{dev}} \frac{\partial \tilde{M}(f_{w_T}(x),y)}{\partial w_T}$. Since w_t is already obtained and updated during the training process of the student model, w_t is no longer updated while training the teacher model. It is used for calculating the derivative with respect to θ. θ is updated with calculating $d\theta$ cumulatively for T steps. The training process of the teacher model will continue until the convergence of θ. The detailed derivation of $d\theta$ and recursive computation of

$\frac{\partial \tilde{M}(f_{w_T}, D_{dev})}{\partial w_t}$ can be seen in [Wu et al., 2018].

Instead of using fixed and well-designed loss function, Wu proposed a gradient based framework of learning dynamic loss function. It has proven to be more efficient in time compared to other optimization approach such as reinforcement learning. Additionally, the framework does not require any human knowledge and achieves significant improvement in NMT and image classification. However, the convergence of the teacher model is usually costly even if the teacher model is light. Because updating the teacher model requires a course of training of the student model from scratch. Further, without leveraging any empirical knowledge may worsen the convergence issue, since the teacher model may wander in parameter space blindly.

2.6 Offensive Language Detection

In this section, we will introduce the offensive language detection task as well as the datasets, and review the literatures about the existing systems for offensive language detection.

2.6.1 SemEval Challenge and Datasets

SemEval-2019 Task 6A (SE19T6A) is one of the subtasks of SemEval-2019 Task 6 (OffensEval-2019) which focuses on identifying and categorizing offensive language in social media. The gold of SE19T6A is to identify offensive and non-offensive contents. Offensiveness is perceived as veiled or direct insults, threats, any form of untargeted profanity, and swear words [Zampieri et al., 2019b]. Each post is assigned one of the two labels: offensive (OFF) and non-offensive (NOT). The challenge is based on a new Offensive Language Identification Dataset (OLID) [Zampieri et al., 2019b] containing 14100 English twitter posts.

SemEval-2020 Task 12A (SE20T12A) is one of the subtasks of SemEval-2020 Task 12 (OffensEval-2020) [Zampieri et al., 2020]. It shares the same purpose as OffensEval-2019 and use the same taxonomy of the OLID schema. Compared with OffensEval-2019, OffensEval-2020 prepares tasks for five languages and extend an additional Semi-Supervised Offensive Language Identification Dataset (SOLID) with more than 9M English twitter posts [Rosenthal et al., 2020a]. In our thesis, we solely focus on the English data.

OLID (SemEval 2019 data) is a large-scale and English dataset of tweets with high quality human annotations, containing 14100 tweets in total. The tweets were collected via Twitter's API using keyword searching. The keywords that are likely to spread offensive speech were selected. Given that most of the tweets in real world are not offensive, the collation of OLID maintains the unbalanced distribution. The full list is shown in Table 2 with the proportion of offensive tweets in each keyword. For reminder, all user mentions and URLs are substituted by placeholders for anonymization. As

Keywords	Offensive tweets (%)
you are	21.0
she is	26.6
@BreitBartNews	31.6
he is	32.4
gun control	34.7
filter:*unsafe*	58.9
conservatives	23.2
antifa	26.7
MAGA	27.7
liberals	38.0

Table 2: Selected keywords for retrieving tweets from Twitter's API and the percentage of offensive tweets for each keyword. Note that *@BreitBartNews* includes the tweets sent by *@BreitBartNews* account and filter:*unsafe* corresponds to tweets that were flagged as unsafe by Twitter.

they consider political contents tend to be more offensive, eventually, half of the tweets in OLID are retrieved from political keywords. They divided OLID into a training set and a test set. The label distribution is shown in Table 3.

Label	Training	Test	Total
NOT	8840	620	9460
OFF	4400	240	4640
Total	13240	860	14100

Table 3: Label distribution of OLID training and test set.

SOLID (SemEval 2020 data) contains over 9 million English tweets that were also collected via Twitter's API using keyword searching. The keywords selected for SOLID are different from OLID's. In order to collect domain independent tweets, a

set of English stop words were used for keywords, such as *the, of, and, to*, etc. The resulting tweets were labeled by four machine learning systems training on OLID following democratic co-training set-up [Rosenthal et al., 2020b]. Democratic co-training is a semi-supervised technique that leverages the models trained on supervised data to automatically tag silver scores to unsupervised data. For SOLID, four machine learning systems are utilized for democratic co-training, they are: PMI [Turney and Littman, 2003], Fast-Text [Joulin et al., 2017], LSTM, and BERT. The average prediction confidence for positive class (OFF) are taken as the silver scores, scoring from 0 to 1. The higher a tweet scores, the stronger confidence for it to be truly offensive. Apart from the averaged confidence scores, standard deviation is also provided. Eventually, the large-scale silver standards are created.

We inspect the confidence score distribution of SOLID as shown in Figure 4 The

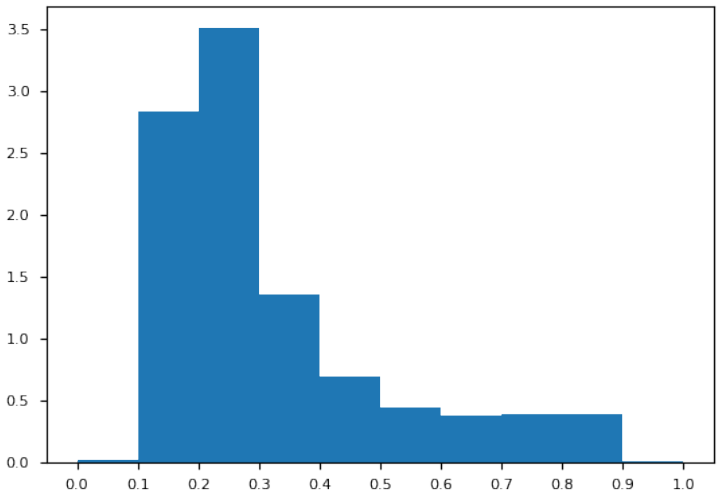

Figure 4: Histogram showing the confidence score distribution of SOLID. The x axis is divided evenly into 10 bins. Each of them contains the samples whose scores fall into the interval. The y axis is the probability density in which each bin will display the bin's raw count divided by the total number of counts and the bin width.

tweets scoring from 0.1 to 0.4 account for the nearly 80% of the data while the extreme tweets are rare.

For testing, 1397 potential offensive tweets are selected following the above procedure, and then they are further manually annotated resulting in 1090 offensive tweets. Afterwards, another 2500 potential non-offensive tweets are also selected without

further manual annotation. Assuming the 0.5 as the threshold between offensive and non-offensive, the label distribution of SOLID are shown in Table 4.

Label	Training	Test	Total
NOT	>7.6M	1090	-
OFF	>1.4M	2807	-
Total	>9M	3897	>9M

Table 4: Label distribution of SOLID training and test set, assuming that the silver scores above 0.5 are labeled as OFF and below 0.5 are NOT.

A few examples of tweets from OLID and SOLID are provided in Table 5. We can see that, in SOLID, the higher score indicates the stronger offensive signal (swear words and profanity). The samples close to 0 show clearly non-offensive signal. Those in the middle, tend to be vague and ambiguous. As for OLID, it corresponds to what we discussed on the nature of offensive language section. For the rest of this section, we will introduce the existing offensive language detection systems.

Dataset	Label/Score	Tweet
SOLID	0.14	*@USER It is haha one more for the road: Tyler Hansbrother*
SOLID	0.44	*@USER most of her fits are straight fire, except the coachella furry pants* 😌 *(2)*
SOLID	0.51	*@USER @USER How? Is he not potbellied or a midget?*
SOLID	0.57	*"I felt personally attacked! My dad was a fat man when I was a little boy!"*
SOLID	0.69	*i really fucking miss having someone to talk to all day*
SOLID	0.88	*@USER Popeye's spicy chicken on some waffles = the shit*
OLID	OFF	*@USER YESSSSSHHHHH who else call you shitgull if it were not me or ryme* 😚
OLID	OFF	*@USER I want to kiss your ass.*
OLID	NOT	*Are you fucking serious? URL*
OLID	NOT	*@USER @USER I don't think he's guilty....this is what you leftwingers do...make up sexual assault claims about conservatives.*

Table 5: Real tweet examples from OLID and SOLID

2.6.2 Existing Systems for Offensive Language Detection

NULI, developed by [Liu et al., 2019a] ranks 1st place in SE19T6A and achieved the SOTA results. NULI outperforms the second place system by 1.4% in F1 score, which is the largest gap among the top systems. Their system is based on BERT. What

distinguishes them from others is the preprocessing. Particularly, they mapped emojis to substituted meaningful phrases using an open project on github. Since BERT is not designed for emojis, emoji substitution may enrich the text features for BERT. It is popular to add a '#' to the beginning of an unbroken text to create a hashtag in tweets. They employed another open project to tokenize the hashtag and eliminated the '#' (e.g. *#GUNCONTROL* → *GUN CONTROL*). At last, they substituted *URL* mentions with *http* and limited the occurrence of repetitive *@USER* mentions to 3. They stratified split the OLID into 9:1.5 as training and development set and fine tuned the BERT model for 3 epochs. The F1 score on the development set and official test set is 78.3% and 82.9% respectively.

Nikolov-Radivchev is developed by [Nikolov and Radivchev, 2019] ranking the 2nd place in SE19T6A. For preprocessing, they excluded all @ and # symbols and tokenized the hashtags as NULI. The data was split into a training and development set in a ratio of 10:1. Afterward, They fine tuned BERT-large-uncased model and it achieves 78.1% and 81.5% of F1 score with development data and official test data respectively.

UM-IU@LING, developed by [Zhu et al., 2019] ranks 3rd place in SE19T6A. No specific preprocessing is conducted. They fine tune the BERT-base-uncased model on OLID and use the output *[CLS]* token embedding as the input for the classification layer. Binary CE loss is used for loss function. They take the small trial dataset as their development set which is available during the competition. The reported F1 score on development set and official test data is 83.9% and 81.4% respectively.

MIDAS, built by [Mahata et al., 2019] qualifies the fourth place in SE19T6A. They takes a careful look at OLID raw data and dispute that about 4% of the tweets are seemingly mislabeled. The issue is more often in NOT label tweets (e.g. *ANGELINA IS SO FUNNY AT RHE WRONG TIMES IMNGONNA SHOOT THIS BITCH UPPDOALS* Label: NOT). Experiments shows that the system benefits from removing these samples. Besides, a few common swear words and their variants are normalized to unified forms respectively. '#' symbols are removed and all hashtags are tokenized as NULI. Eventually, they submitted an ensemble system of CNN and bidirectional LSTM with attention and bidirectional LSTM followed by bidirectional gated recurrent unit (GRU) achieving F1 score of 80.7% with official test set.

As examined by [Zampieri et al., 2019b], 7 teams out of the top 10 fine tune the

BERT in SE19T6A and we have reviewed 4 systems in detail. A summary for the top 6 systems are listed in Table 6. We can see from their experiments, the official test set is simpler than the training set as models increase by approximately 3% in testing. Most of the systems leverage transformer based structure, and the performance gaps are trivial. Slight changes in preprocessing may lead to the change of ranking. Besides, the improvement brought by the model ensemble is also subtle. Yet, it makes the system bulky and increases inference time. For the rest of this section, we will review the systems in SE20T12A.

System	Model	F1 score(%)
NULI	BERT*	82.9
Nikolov-Radivchev	BERT-large-uncased	81.5
UM-IU@LING	BERT-base-uncased	81.4
Embeddia [Pelicon et al., 2019]	BERT-large-uncased	80.8
MIDAS	CNN+BiLSTM+LSTM,GRU	80.7
BNU-HKBU [Wu et al., 2019]	BERT-base-uncased	80.6

* The type of BERT is not reported

Table 6: Summary of top 6 systems for SE19T6A. The table is sorted according to the ranking of SE19T6A in descending order. Some systems not discussed herein, come along with citations since they use similar techniques.

UHH-LT is the 1st ranking system in SE20T12A [Wiedemann et al., 2019]. Specifically, they perform fine-tuning on BERT and some BERT variants models with supervised dataset OLID. In addition, the models are further trained via the MLM task on SOLID for the sake of domain adaptation. Since domain adaptation requires unsupervised data, they avert handling the silver standard. Regarding the hardware limitations, 5% of SOLID data (436k) are randomly sampled and are used for MLM training. For preprocessing, *@USER* and *URL* mentions are removed beforehand. In the competition, an ensemble of RoBERTa-based models was submitted in which the further MLM trained model is fine tuned 10 times using 90% of OLID data leaving the rest for validation. The F1 score on official test data of SE20T12A is 92.04%.

Galileo qualifies the 2nd ranking in SE20T12A [Wang et al., 2020]. Galileo aims to build a multilingual model for offensive language detection. The work is built on XML-R [Conneau et al., 2020] which is a RoBERTa based cross lingual language model pre-trained on Common Crawl dataset in over 100 languages. They add a

classification layer over XML-R and fine tune XLM-R with OLID data, for English. Galileo achieves high performance on OffensEval-2020 for all languages, and the F1 score for English official test data is 92.0%.

Rouges [Dadu and Pant, 2020] utilizes XML-R as [Wang et al., 2020] and ranks 3rd. They fine tune the XML-R model sequentially on supervised data. OLID is split into 9:1 as training and validation set. The reported F1 score on validation data and official test data of SE20T12A is 70.9% and 91.9% respectively.

GUIR [Sotudeh et al., 2020] ranks fourth place in SE20T12A. They preprocess the data similar as [Liu et al., 2019a] did in NULI and further train the BERT-base model with MLM and next sentence prediction objectives in unsupervised manner for the sake of domain adaptation. They report the F1 score both on SE19T6A (82.8%) and SE20T12A (91.7%).

KS@LTH [Socha, 2020] is the 5th ranking system. They first fine tune the RoBERTa-large model with OLID supervised data. In order to use the semi-supervised data, they manage to sample 400,000 tweets from SOLID based on the silver standard score as weight. Afterward, the 400,000 tweet samples are further labeled using the fine tuned RoBERTa model. 20,000 tweets with the highest score are labeled as OFF, while the 20,000 lowest are labeled as NOT. In this way, 40,000 further labeled tweets are joined into the training set. The resulting model achieves 91.5 F1 score on SE20T12A test data. However, they submitted another RoBERTa model for SE20T12A, because they found that rather than using the silver standard, fine tuning the RoBERTa on Davidson (an open dataset for hate speech and offensive language) [Davidson et al., 2017]. This outperforms the previous by 0.2%.

Hitachi [Ravikiran et al., 2020] ranks 34th in SE20T12A with F1 score on official test data of 0.909. To utilize the silver standard, they introduce a sampling algorithm to access the trustworthy tweets from SOLID. Specifically, they retrieve the tweets whose confidence standard deviation fall into a predetermined range. this reduces the amount of tweets to 2.4M. Afterward, this filtered subset is combined with a auxiliary dataset from OffensEval-2019 subtask B in which all tweets ($\approx 0.2M$) are offensive and can also be seen in SOLID. This results in less than 2.6M unbalanced subset, in which tweets from the majority class are randomly removed to release the class imbalance issue. The final F1 score on SE20T12A test data is 90.9%.

Although the test set of SOLID is larger than OLID's, the results indicate SOLID

test data is much easier. As SE20T12A also provides training data for 5 languages, Galileo and Rouges leverage all of them and fine tune on RoBERTa based cross lingual pre-trained language mode and yield high performance. Despite this, the gains brought by increasing model size are still subtle. With domain adaptation and BERT base model, GUIR is able to surpass many larger size models. Although domain adaptation leverages SOLID data, it is an unsupervised approach and does not use the silver standard. As we know that SOLID is a set of random tweets collected from Twitter, domain adaptation is nothing more than adapting in Twitter language. But it is still undeniable that it enhances the model's representation ability. The drawbacks of domain adaptation would be relatively costly in time. KS@LTH uses their own massive re-labeled SOLID data and discards the silver standard labels. Hitachi resamples the SOLID using the silver standard but underperforming. To the best of our knowledge, overall, few teams leverage the whole SOLID due to its massive volume, hence, taking subsets is the common approach. Table 7 shows a system summary on SE20T12A.

System	Model	F1 score(%)	Use of SOLID	Use of Silver Standard
UHH-LT	RoBERTa-base	92.0	Domain adaptation	False
Galileo	XML-R	92.0	-	False
Rouges	XML-R	91.9	-	False
GUIR	BERT-base	91.7	Domain adaptation	False
KS@LTH	RoBERTa-large	91.7	Further re-labeling	False
Kungfupanda [Dai et al., 2020]	BERT-based	91.5	-	False
Hitachi	BERT-base-uncased	90.9	Selecting a subset	True

Table 7: System summary for SE20T12A. The table is sorted according to the ranking of SE20T12A in descending order. Some systems not discussed herein, come along with citations.

Two team top teams out of seven use SOLID for domain adaptation, with three of them discarding it, while the one team re-label the silver standard with their own system and only last team truly exploited the raw silver standard. To the best of our knowledge, the silver standard is rarely exploited and the approaches to properly exploit SOLID are scarce.

Chapter 3

Experiments

Taking the SemEval shared tasks (SE19T6A and SE20T12A), the supervised offensive language identification dataset OLID and semi-supervised dataset SOLID are used. The overly large size of SOLID and its questionable annotations impede lots of studies truly using the dataset. Addressing these problems, two practicable ways using the SOLID are investigated. Class parameters leverage the curriculum learning and enable a model to assign weights to samples of different difficulties, while dynamic loss functions offer more flexibility in controlling difficulties. The two ways are inspired based on the preliminary findings on selective fine tuning over OLID and SOLID. Before introducing the experiments, the two datasets are first compared from linguistics aspects, in order to gain insights of the two tasks and the datasets.

3.1 Comparing Datasets

This section analyzes the datasets (OLID and SOLID) by providing statistics respectively, and compare them with each other standing on the view of linguistic notions. To be able to operate on SOLID and provide inner analysis, SOLID is divided into 20 folds, namely SOLID-fold, each of which contains 450,000 tweet samples approximately. To show the word token level distribution of SOLID and OLID, the word frequency of the most common words in two SOLID folds and OLID are calculated respectively. ANNIE [Cunningham et al., 2002] is used as the tokenizer and remove the English stop words. The English stop words such as *the, a, of, is, to etc.* could be distracting under this experiment because they are too common in English and are presumed

to be uninformative in representing the semantics without context. Regarding the all-capitalized tweets and the casual spelling, the texts are converted to lower cases. As shown in Figure 5, 6, and 7, the most 25 words are selected in SOLID-fold-4, OLID, and SOLID-fold-7 respectively and calculate the normalized word frequency: $\frac{\text{\# Raw Word Frequency}}{\text{\# Total Number of Tweets}}$. Note that *don* is the tokenized form of *don't*. The x axis is the frequent words and the y axis is the normalized word frequency.

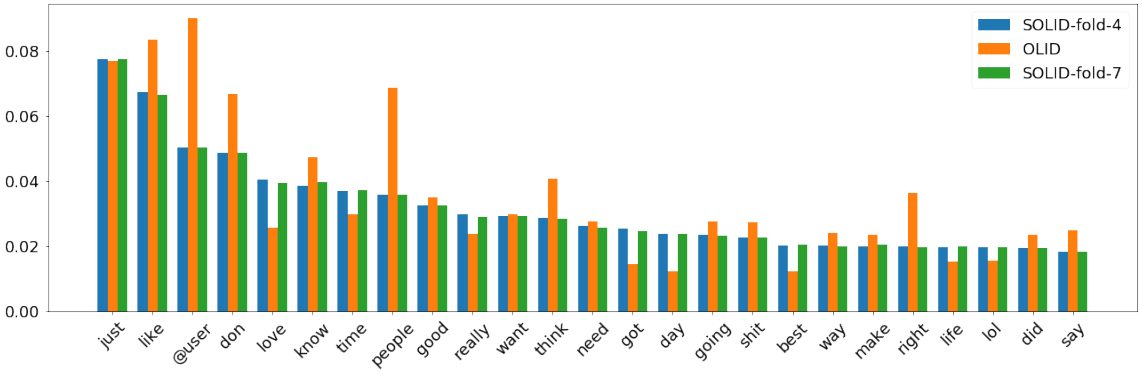

Figure 5: Normalized word frequency of the most frequent words in SOLID-fold-4

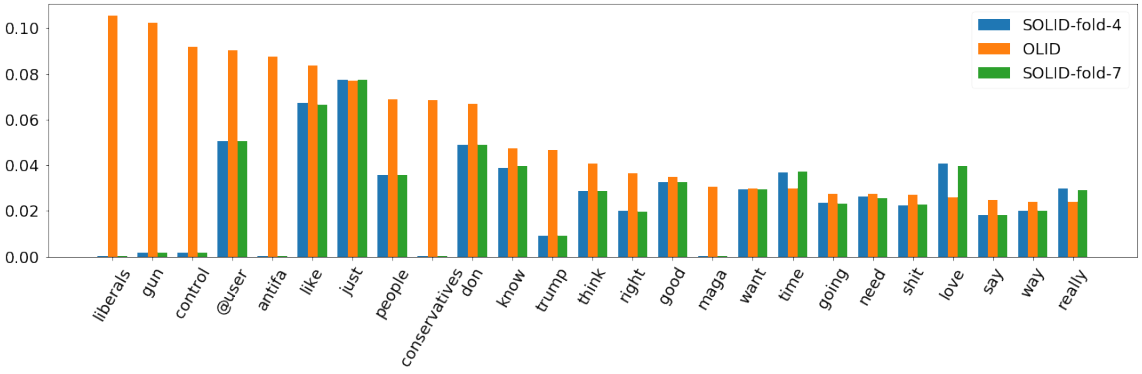

Figure 6: Normalized word frequency of the most frequent words in OLID

We can see from Figure 5 and 7 the top words from SOLID-fold-4 and SOLID-fold-7 are almost identical. The word frequency of the top words in SOLID folds (the blue and green bars) are almost at the same level. However, among these words, the frequency of such as *like*, *@user*, *don*, *people*, *know*, *think*, *right*, in OLID is even higher, It can be infered that utterances in OLID are more opinionated and expressive

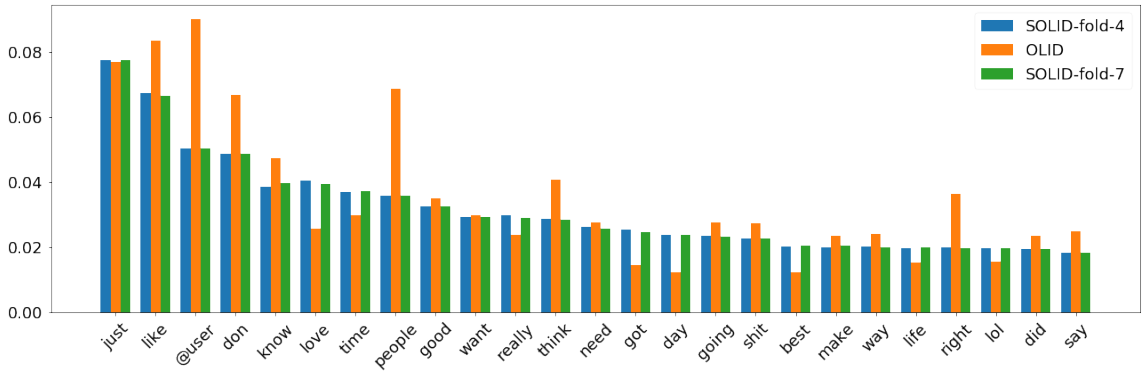

Figure 7: Normalized word frequency of the most frequent words in SOLID-fold-7

than those in SOLID. We also observe that people enjoy adding redundant *@USER* repeatedly for emphasis. The term *right* should be used a lot more in the political context in OLID, as we describe the genre of tweets in OLID in the earlier section.

As for Figure 6, the trending words in OLID are distinct from SOLID's in which the political terms even surpass the common words in daily conversations such as *just* and *like*. However, those political terms rarely occur in SOLID folds, making the huge divergence of the word composition of OLID and SOLID. Interestingly, we notice that *love* occurs less often in OLID.

To inspect more on word level distribution, an offensive word list[1] is employed that is collected by Luis von Ahn's Research Grouping from CMU and includes over 1300 English offensive terms, most of them are profanities. Besides, AFINN [Nielsen, 2011], a manually crafted sentiment lexicon that assigns sentimental scores to 2477 English sentiment-laden words is used. The sentimental score ranges from -5 to 5, the higher score the more positive in sentiment, while 0 indicates neutral. Afterward, the words are divided by their scores and sum up the normalized frequency among the three comparing datasets respectively, and ditto the CMU list. As shown in Figure 8, overall, the sentimental-laden words are more frequent in OLID than SOLID folds and the frequency even doubles in terms of the profanities.

Further, the Spearman rank correlation coefficient is used as a measure of corpora similarity. Spearman rank correlation coefficient, denoted as ρ, is a nonparametric measure of correlation between the rankings of two variables. It assesses how tight

[1] `https://www.cs.cmu.edu/~biglou/resources/`

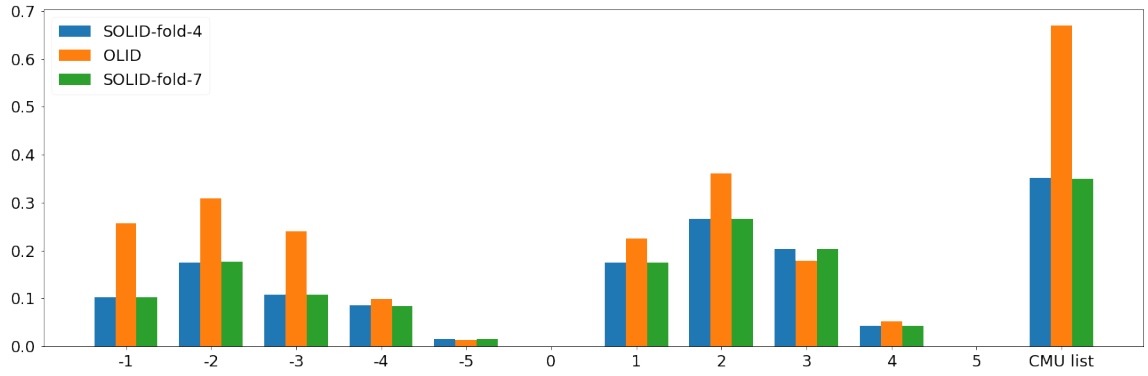

Figure 8: Summing the normalized frequency in the CMU list and words with different AFINN scores.

the relation between two variables can be measured. The value of ρ always ranges from -1 to 1 where the negative value implies the negative correlation. The value of ρ can be computed using equation 20.

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \tag{20}$$

where n denotes the number of observations and d_i represents the difference between two ranking. In our expermients, word ranking is obtained in following steps: given two datasets to operate on,

- union two datasets.
- select the top n common words from the union dataset.
- compute the sorted word count vectors of two datasets respectively.
- for each most common word, the ranking in two datasets namely r_a, r_b are obtained from the word count vectors.
- compute the value of ρ using formula 20 where $d_i = r_a - r_b$

Specifically, five subsets are randomly sampled from SOLID with the same size of OLID, namely SOLID-subset, and then computer the ρ values with OLID respectively. The results are shown in Table 8. In addition, the ρ in between the subsets are also computed, and the averaged score are provided at the last row.

Corpus vs Corpus	top 500	top 320	top 200	top 150
OLID vs SOLID-subset-1	0.590	0.652	0.660	0.675
OLID vs SOLID-subset-2	0.613	0.652	0.659	0.674
OLID vs SOLID-subset-3	0.596	0.651	0.669	0.674
OLID vs SOLID-subset-4	0.620	0.656	0.653	0.665
OLID vs SOLID-subset-5	0.601	0.649	0.663	0.667
Inner SOLID-subsets	0.998	0.998	0.999	0.999

Table 8: ρ values between OLID and SOLID-subsets and in between SOLID-subsets.

The numbers are stable when comparing OLID with SOLID-subsets. It is reasonable that the ρ increases as n decreases, as the common word bucket shrinks, the difference (d_i) between ranks reduces. The word level similarity between OLID and SOLID-subsets are largely divergent compared with the inner ρ values of SOLID-subsets.

Given that a corpus is never a random set of words, natural languages share commonalities in terms of diction or syntax. Above experiments demonstrate the word level distinction of OLID and SOLID. They showcase a highly repetitive nature between the SOLID folds and the varying content between OLID and SOLID at word level. For the rest of this section, we apply the above statistics on higher level linguistic notions: POS and typed dependency, which may give us hints of the syntax differences between SOLID and OLID. We calculate the POS frequency and typed dependency frequency out of OLID and SOLID-folds respectively as shown in Figure 9 and 10.

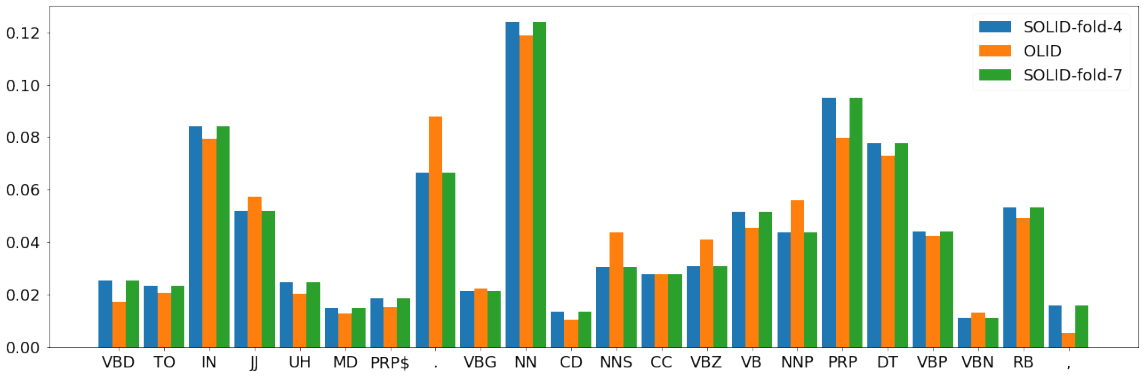

Figure 9: Normalized POS frequency of three datasets

The POS tag and typed dependencies with low frequency are removed from the figure. We notice that the frequency of *VBZ* (present tense, 3rd person singular

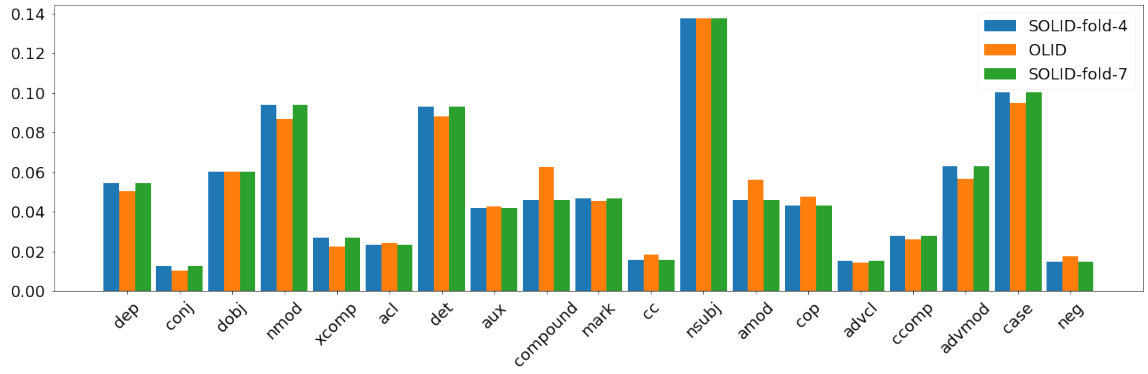

Figure 10: Typed dependency frequency of three datasets

verb), *NNP* (proper noun, singular), *NNS* (plural noun) and *JJ* (adjective) notably surpass that of SOLID-folds, which corresponds with our observation that political terms and sentiment-laden words are more prominent in OLID but less in SOLID. The dependency distribution is less volatile between OLID and SOLID-folds where *compound* (general modifier) and *amod* (adjectival modifier) of OLID slightly pass that of SOLID-folds.

To have a better insight of the offensive language in both OLID and SOLID, we redo the statistics on OFF[2] data and NOT data respectively. As we have seen before, the SOLID-fold-4 is almost identical to SOLID-fold-7 from the perspective of linguistic statistics. To avoid the handling the whole SOLID, and linguistically inspect the difference of offensive language between OLID and SOLID, we combine the OFF data from those folds temporarily in this section for comparison, namely SOLID-OFF and take the NOT data from SOLID-fold-7, namely SOLID-NOT and the OFF data from OLID, namely OLID-OFF. To observe linguistic features of the offensive utterances, we compute the normalized POS frequency and normalized typed dependencies in SOLID-OFF, OLID-OFF and SOLID-NOT respectively. The results are shown in Figure 11 and Figure 12. Interestingly, these two figure resemble Figure 9 and Figure 10, despite the frequency of *NN* in OLID slightly surpass the others. The resemblance indicates that the OFF utterances are offensive in different ways between SOLID and OLID. As we observed before, the frequency of *VBZ*, *NNP*, *NNS* and *JJ* (adjective) in offensive context still surpass that of SOLID. This raise our interest

[2]The labels are defined in section SemEval Challenge and Datasets where OFF stands for offensive data and NOT for non-offensive.

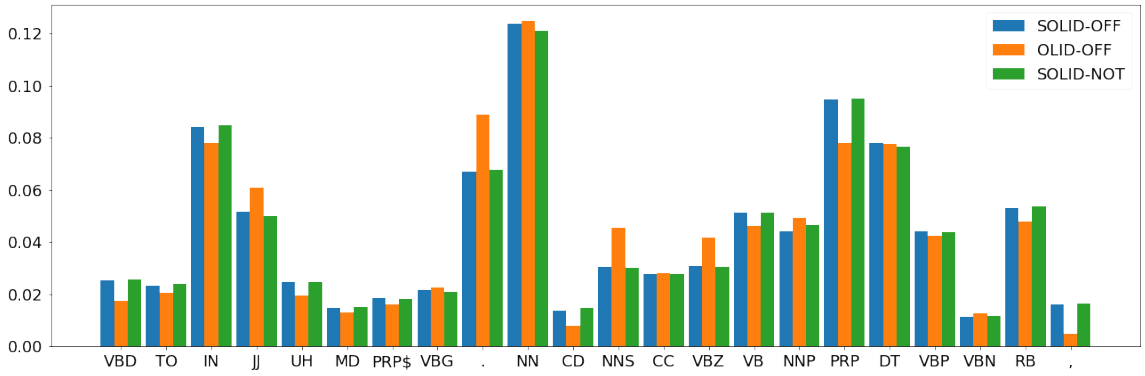

Figure 11: Normalized POS frequency of SOLID-OFF, OLID-OFF and SOLID-NOT

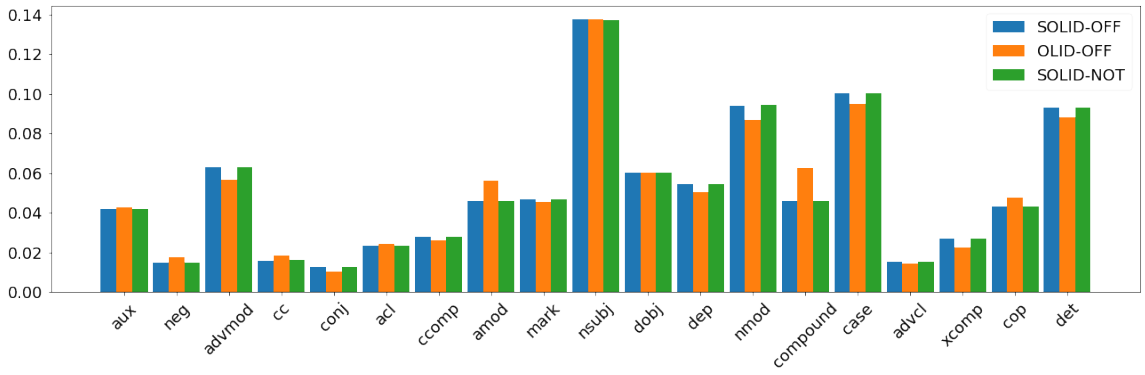

Figure 12: Normalized typed dependency frequency of SOLID-OFF, OLID-OFF and SOLID-NOT

to discover the different offensive diction between the two datasets. Therefore we compare the normalized term frequency of most frequent words among SOLID-OFF and OLID-OFF. The results are shown in Figure 13 and Figure 14. From Figure 13 and Figure 14 we can see the frequency of some profanities are distinct between two SOLID-OFF and OLID-OFF. Their frequencies in SOLID-NOT are significantly higher than those of OLID-OFF, which may suggest that user prefer cussing straightforward with profanities in SOLID, however, in OLID, user tend to be more obscured and veiled.

In conclusion, the high overlap of most of the above statistics between the folds of SOLID showcases the repetition nature of SOLID. Comparing SOLID and OLID, they are not only distinct from the content of tweets, but also from the phrasing and

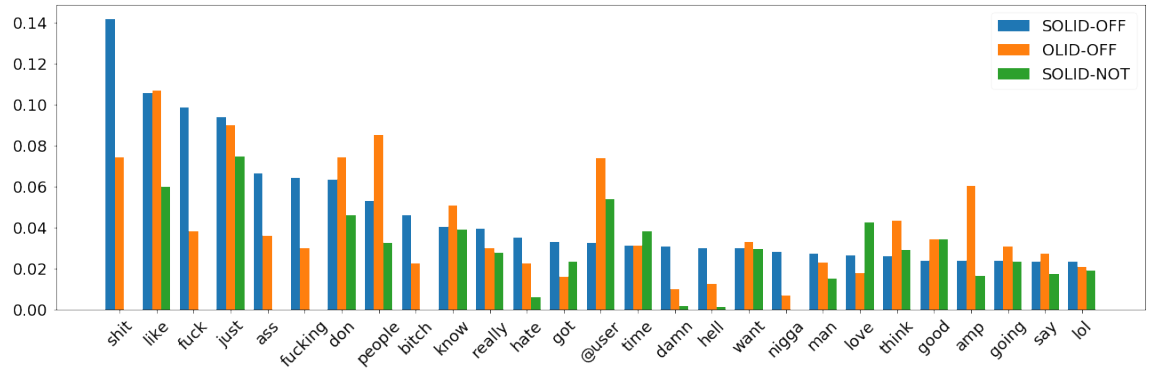

Figure 13: Normalized term frequency of frequent words in SOLID-OFF

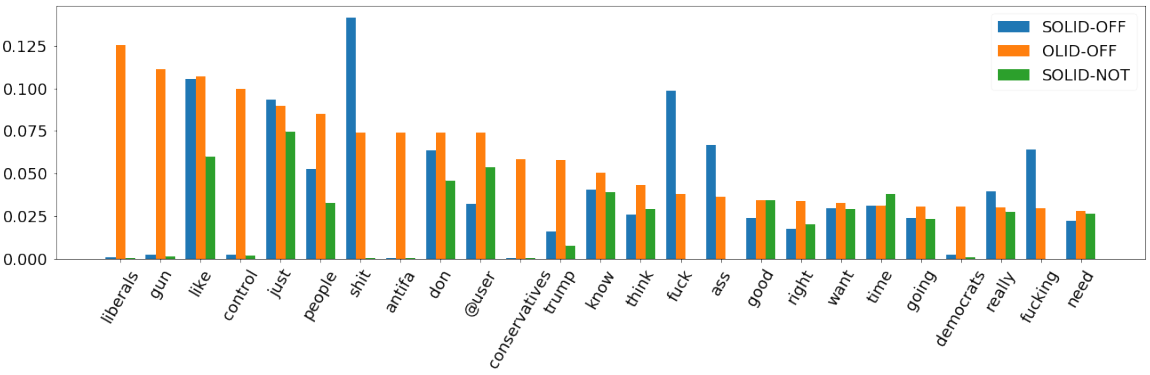

Figure 14: Normalized term frequency of frequent words in OLID-OFF

the way being offensive. More political terms and sentiment laden words in OLID compared to SOLID indicates that users in OLID tend to be expressive. However, when comparing the OLID-OFF and SOLID-OFF, the SOLID is the one who prefers cussing straightforward; OLID, in contrast, is less direct. This could serve as an evidence of the simplicity of SOLID.

3.2 Baseline Study

To get the baselines, the training samples in OLID are split into a training and development set in a ratio of 8:2, which results in 2648 samples in the development set, namely OLID development set, containing 1763 OFF labels and 885 NOT labels. We use this development set across our experiments, and provide the final result in

official test data at the end.

A series of vanilla models: BERT, RoBERTa, DistilBERT, DistilRoBERTa, HateBERT, BiLSTM, and CNN are trained with OLID and validate them on the OLID development set. The results are presented in Table 9. The hyper-parameters are selected via random search based on prior experience and the Transformer based model are fine tuned for 1 epoch with learning rate 1e-5, while the CNN and BiLSTM are trained for 3 epochs with learning rate 2e-4. The Adam optimizer [Kingma and Ba, 2014] are used across our experiments, unless another specified. The use GLoVe word embeddings as the input for BiLSTM and CNN and for Transformer based models, the *[CLS]* token embedding is extracted from the last hidden states as the input of the classification layer. This thesis refers this as the default way to fine tune the Transformer based models in the rest of the experiments. Following the hyper-parameters recommended in [Caselli et al., 2021], that is learning rate of 1e-5 and batch size of 32, HateBERT is fine tuned and is then validated. From the baselines in Table 9, we see that the

Model	F1-1	Macro F1
RoBERTa	0.716	0.788
BERT-base	0.716	0.787
DistilRoBERTa	0.715	0.783
DistilBERT	0.707	0.783
HateBERT	0.686	0.759
BiLSTM	0.636	0.744
CNN	0.631	0.739

Table 9: The performances of baseline models. The evaluation metric of the OffensEval is used, which is macro-averaged F1 (Macro F1). For better presentation, F1 score with respect to positive class (offensive class) is added, namely F1-1. The results are ranked by F1 score in descending order.

Transformer based language models notably superior to training a neural network from scratch. They require less training epochs and yield higher results. The performances of using BERT or RoBERTa or their distilled version are not distant. The macro F1 and F1-1 does not benefit largely from the increase of model capacity. The HateBERT loses its competitiveness. This may caused by the incompatibility of offensive tweet language and the abusive Reddit language that the HateBERT is specialized. Thereby the DistilBERT model is used as our default pre-trained model for the rest of our experiments since it yields the competitive results to the larger pre-trained models

with lite model size and the fine tuning and inference are faster. In addition, without any extra efforts, the RoBERTa baseline hit over 78% in F1, surpassing the CNN by 5%. However, not to mention the model's sizes of them are tremendously distinct, 5% improvement under the great pre-trained effort of RoBERTa also seems insignificant. The closeness between the baselines showcases that the visibility of improvement is small and difficult.

3.3 Selective fine tuning on SOLID

This thesis employs continual learning on our tasks, in which a model is first fine tuned on a dataset, after which the model is further fine tuned on another in domain heterogeneous dataset. To address the catastrophic forgetting problem, the idea of mimicking the data distribution [Kemker et al., 2017, Shin et al., 2017] is employed in a simplified manner. That is, to approximate the label distribution of OLID, down-sampling on the overwhelming class data is applied to SOLID. The down-sampling is performed when subsets are sampled from SOLID. Likewise, we borrow the aspects from curriculum learning to mitigate the catastrophic forgetting as shown in [Greco et al., 2019]. Specifically, the models are selectively fine tuned with different subsets of multiple difficulty levels. The experiment results are then investigated and analyzed. In addition, we will also present experiments showing the limitation of model capacity by conducting a long term fine tuning and discuss to what extent that models gain from long term fine tuning.

It is a challenge to exploit the SOLID, since its size is rather large and the nature of the silver standard. When it comes to BERT fine-tuning, it is estimated to run over weeks on a common CPU machine. Sampling a subset from it is a common approach among the teams who took part in the SE20T12A. As it investigated before, two top teams out of seven use it for domain adaptation, with three of them discarding it, while the one team re-label the silver standard with their system but rarely properly exploit SOLID. Addressing this issue, we make the first attempt to investigate the potential of SOLID by selectively leveraging the subsets of SOLID with various sizes and content, taking the idea from continual learning. Beside, we manage to fine tune a pre-trained model with different SOLID folds incrementally. Overall, in this section, we present:

- selectively further fine tuning a pre-trained language model with different SOLID subsets separately.
- fine tuning with different SOLID folds incrementally.

In order to reduce the model complexity, we decide to use DistilBERT model as our default pre-trained language model across our experiments.

Selective fine tuning the DistilBERT model is fine tuned with OLID for one epoch, then is further fine tuned with a small subset of SOLID for another epoch. Since it is fine tuned successively, the weights are shared and the same hyper-parameters settings are used across two epochs. The complementary subsets are sampled based on simple criterions (the coverage of confidence scores) which are described in the caption of Table 11. And a summary of these subsets is in Table 10. Note that we preserve the distribution of OLID while sampling the subsets, by means of down-sampling the majority class (non-offensive class). Results are shown in Table 11 and Table 12. The results are averaged over at least 5 runs, if no specific mention.

Subset	Size	Down-sampling	Balanced
SOLID-TH46	14100	Yes	No
SOLID-TH55	14100	Yes	No
SOLID-TH55b	14100	Yes	Yes
SOLID-TH55f	450,000	Yes	No
SOLID-{1...5}	14100	No	No

Table 10: Description of some subsets. The subsets are sampled based on conditions described in the caption of Table 11.

As shown in Table 11, very little improvement is made during the second epoch without SOLID corpora. Both SOLID-4 and SOLID-5 outperform other random subsets whereas F1 scores even descent in SOLID-2, 3. None of a random subset rises in F1 measure with respect to positive class. We speculate the reason may be due to the severe data unbalance of SOLID corpora where negative samples (non-offensive class) accounts for higher proportion than that of OLID. As shown in Table 8, SOLID-4, 5 are more distant from SOLID, compared to others. Increasing data variety could benefit further training.

For conditional subsets, apart from setting thresholds, they are sampled following the sample distribution of OLID, which amplifies the contribution of positive class. As

Fine tune on		First fine tuning	Second fine tuning	Difference	Ranking
OLID	Macro F1	.782	.781	-.1%	#7
	F1-1	.705	.705	0%	
OLID, SOLID-1	Macro F1		.780	+.1%	#9
	F1-1		.692	-.7%	
OLID, SOLID-2	Macro F1		.775	-.4%	#12
	F1-1		.682	-2.23%	
OLID, SOLID-3	Macro F1		.777	-.2%	#11
	F1-1		.687	-1.96%	
OLID, SOLID-4	Macro F1		.785	+.6%	#3
	F1-1		.704	-.12%	
OLID, SOLID-5	Macro F1		.786	+.68%	#2
	F1-1		.702	-.32%	
OLID, SOLID-TH55	Macro F1		.788	+.88%	#1
	F1-1		.718	+1.25%	
OLID, SOLID-TH55b	Macro F1		.781	+.2%	#7
	F1-1	.779	.712	+.7%	
OLID, SOLID-TH56	Macro F1	.705	.780	+.1%	#9
	F1-1		.693	-1.2%	
OLID, SOLID-TH45	Macro F1		.784	+.54%	#5
	F1-1		.713	+.84%	
OLID, SOLID-TH46	Macro F1		.785	+.64%	#3
	F1-1		.700	-.46%	
OLID, SOLID-TH37	Macro F1		.782	+.32%	#6
	F1-1		.705	0%	
OLID, SOLID-TH38	Macro F1		.746	-3.32%	#14
	F1-1		.627	-7.78%	
OLID, SOLID-TH28	Macro F1		.749	-3.0%	#13
	F1-1		.632	-7.3%	
OLID, SOLID-TH19	Macro F1		.738	-4.1%	#15
	F1-1		.618	-8.7%	

Table 11: Results using different complementary subsets to further train the DistilBERT model. The first epoch is trained on OLID solely and is shared across all experiments as the base tuned model. Note that all subsets above are equal size. Unlisted hyper-parameters are the same across these experiments. The first line sets a baseline where no complementary subset engages in. Additionally, for better comparison, the baseline results are averaged on a dozen runs. SOLID-1 to 5 refer to the SOLID-subsets mentioned at Table 8 respectively. The two digits with the TH indicates a score threshold while sampling a subset, where any confidence score satisfies the conditions: first digit $> Score * 10$ and $Score * 10 >$ second digit. (e.g. SOLID-TH37 indicates this subset contains samples of which scores satisfy $0.3 > score$ and $score > 0.7$.) 'b' denotes a balanced subset where ratio between classes is closed to 1. For clarity, the subset from SOLID-1 to 5 are named random subsets; the subsets with the TH modifier conditional subsets.

Fine tune on		First fine tuning	Second fine tuning	Difference (%)	std.
OLID	Macro F1		.780	+.1	.410
	F1-1		.705	0	.004
OLID, SOLID-TH3558	Macro F1		.794	+1.32	.003
	F1-1		.723	+1.57	.003
OLID, SOLID-TH2558	Macro F1		.792	+1.30	.002
	F1-1		.720	+1.54	.002
OLID, SOLID-TH2557	Macro F1		.792	+1.26	.002
	F1-1	.779	.717	+1.22	.002
OLID, SOLID-TH3468	Macro F1	.705	.789	+1	.002
	F1-1		.710	+.5	.002
OLID, SOLID-TH3557	Macro F1		.786	+.72	.003
	F1-1		.703	-.18	.005
OLID, SOLID-TH2468	Macro F1		.782	+.34	.006
	F1-1		.693	-1.18	.001
OLID, SOLID-TH4556	Macro F1		.770	+.12	.002
	F1-1		.712	+.68	.001

Table 12: Further experiments based on Table 11. In particular, the four digits are used as an identifier where the first two digits and the other two describe two conditions mentioned in Table 11 respectively. (e.g. TH2468 denotes the criterions scoring between 0.2 and 0.4 and between 0.6 and 0.8). Other notations stick to Table 11. The results are ranked by F1 score in descending order.

a reminder, while referencing a subset, the SOLID prefix is omitted for clarity. Among the experiments in Table 11, TH55 strikes the most considerable increase during the second epoch. Neither TH45 nor TH56 surpasses TH55, such that we owe it to the significant contribution of middling samples, especially those barely above .5.

Subset TH55 can be also viewed as a random subset without any criterion, except rectifying the distribution. Comparing it with random subsets, we learn that it is beneficial to uplifting the proportion of positive samples. Nevertheless, the limited improvement on F1-1 of theTH55b subset indicates that a balanced subset does not guarantee a relatively large increase in positive class. Preserving skewness is helpful for our task.

Interestingly, extreme samples such as TH28, TH38, and TH19 deteriorate the performance largely. TH28 gains slight improvement for the negative class, which compensate for the loss of positive prediction; TH19 does not bring any benefits. Consequently, we consider excluding the polarized samples. Corresponding experiments will be provided later.

Consistent drop of macro F1 is observed as the criterion range shrinks. TH55 outperforms both TH46 and TH37 largely, and ditto to TH45, TH56 but relatively

smaller in scale. These underperforming subsets exclude the middling samples while increasing the proportion of extreme samples. This behaviour may imply the importance of middling sample and the possible adverse effect of extreme sample.

Further in Table 12, aware of the impacts of extreme cases, we exclude them by adjusting the criterion range from small and intensive to large and relaxing. Meanwhile, we investigate the impact on excluding the middling cases, especially the tweets scoring between 0.4 and 0.6. Overall, further fine tuning DistilBERT with SOLID excluding the extreme cases is superior to feeding raw SOLID data. Comparing TH2468, TH3468 with TH2558, TH3558, we see more improvement is made if the middling samples are included. This goes against our general perceptions that middling cases trivially contribute to training a classifier for they usually carry vague and sometimes ambiguous signals to the system. In this sense, the middling cases are referred as hard cases while on the contrary, the extreme cases are easy cases with clear meanings that are strong offensive or totally non-offensive signal. However, in the offensive identification task, while further fine tuning a language model, we encourage exposing the model to these middling cases. When it comes to the extreme cases, the system attains the highest macro F1 when excluding the samples above 0.8 and below 0.3 (TH3558). Results shown in Table 12 further justifies our conjecture drawing from Table 11.

Additionally, the experiments are extended on representative folds shown in 13. The results are increased by varying degrees because of abundant training samples. TH3558 still yields the best performance and less volatility among the others. Apparently, excluding the extreme cases benefits the system and preserving the middling cases increase the model robustness.

Incremental fine tuning In this section, we will explore the potential of using SOLID corpora. We take the whole SOLID corpus as training set and validate the model performance on the OLID training set which consists of 14100 annotated samples. Firstly, the corpora is divided into 20 folds where each fold contains 450,000 samples approximately. We design three training strategies namely AC, ER and SP, each of which results in two curves shown in Figure 15, representing the macro F1 and F1 score w.r.t positive class (F1-1).

- **AC**: the folds are trained on the top of each other cumulatively.
- **ER**: the extreme samples of each fold are removed. Then train cumulatively as AC.

Second fine tune on		First fine tuning	Second fine tuning	Difference (%)	std.
OLID	Macro F1	.779 .705	.780	+.1	.410
	F1-1		.705	0	.004
SOLID-TH3558f	Macro F1		.805	+2.63	.002
	F1-1		.733	+2.81	.005
SOLID-TH55f	Macro F1		.799	+1.99	.004
	F1-1		.720	+1.53	.009
SOLID-TH46f	Macro F1		.796	+1.7	.005
	F1-1		.718	+1.3	.009
SOLID-TH3468f	Macro F1		.800	+2.10	.003
	F1-1		.728	+2.25	.009

Table 13: Averaging performances on representative folds. SOLID are first filtered under different criterions. Instead of taking a subset from it, all qualified samples are divided into folds where each fold consists of nearly 300,000 samples. Regarding the time consuming issue, we randomly pick fine tune our model with 5 of the folds separately and average the results. For conciseness, some unnecessary notations are omitted. The standard deviations are provided to inspect the robustness.

- **SP**: the folds are trained separately and independent of each other.

Whatever strategy is used, we validate our models once a fold finishes training, which results in performances for each time step. Note that we train for only one epoch for each fold. For AC and ER, the model generated at the last time step, is trained on the whole SOLID corpus. To verify our analysis of influence of extreme samples, we remove samples scoring above 0.8 or below 0.2 in each fold for ER.

As illustrated in Figure 15, Strategy SP is underperforming at any fold and shows a lot more volatile than AC and ER. The SP serves as a baseline of fine tuning DistilBERT on the folds of SOLID. In general, there is a rising trend among AC and ER curves, despite some fluctuations. However, feeding such massive data does not yield a remarkable performance. The bonus of data volume is very limited: the macro F1 increases merely about 1% training with 9 millon data. This can be explained by the phenomenon of capacity saturation in which the representation ability of the model is limited and the trade-off between retaining learned knowledge and obtaining the new is inevitable. We also find that for macro F1, in the early stage of training, the AC surpasses ER strategy till the fold 7, after which they intertwine and it is hard to tell which strategy is better. However, the F1-1 for ER in fold 1 outperforms the AC but it drops dramatically in later folds. We speculate that in the first fold, the model preserves the memory of the original language model, hence, it benefits from the removal of extreme cases. When it comes to the later folds, due to the catastrophic

forgetting and lack of extreme samples, it loses the memory of those strong signals of offensive features. After incremental learning of a large scale of data, the model regains those feature signals and catches up with the performance of AC.

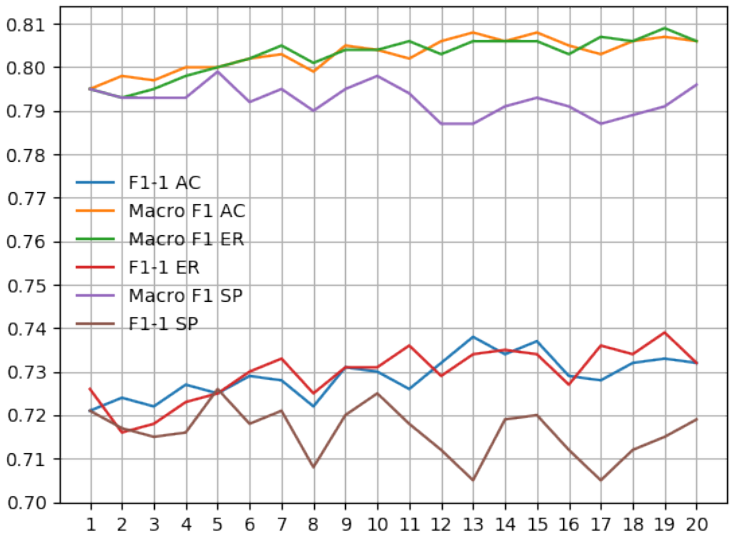

Figure 15: F1 values computing on OLID training set. For AC and ER, the model validated at time step i is trained on the top of folds k ($k <= i; k, i \in \mathbb{Z}^+$). ER indicates that the extreme samples are excluded for each fold. SP denotes the baselines where each fold is trained individually.

The incremental fine tuning experiments showcases the limitation of the using the whole SOLID. It fails to guarantee an apparent improvement. On the contrary, it may suffer from overfitting due to the repetitive nature and the corrupted annotation of SOLID. Therefore, it worths nothing to use the whole SOLID dataset for the trivial improvement.

In conclusion, we find it beneficial to remove the extreme cases when further fine tuning our model, because the extreme cases (easy, clear cases) may distract the preliminarily fine tuned model and impedes the model from further learning. When fine tuning from scratch, the extreme samples could to some extent not interfere the learning process since they often present straightforward strong signals.The middling cases (hard, ambiguous cases) can not be neglected. Since they are easy to flip between offensive and non-offensive, they are tricky but are more informative. What is discovered from the selective fine tuning leads to the automatic curriculum learning

based on the difficulty of samples. It worth further investigation on the how different difficulty samples impact the learner.

3.4 Curriculum Learning on SOLID

This section further leverages the silver standard and investigate the impact of extreme samples and middling samples standing on the view of curriculum learning (CL). A dynamic CL manner is implemented: class parameters [Saxena et al., 2019] in our system and design a series of experiments with the purposes of

1. fitting the semi-supervised data using class parameters;
2. unveiling the effects of applying class parameters on semi-supervised data;
3. getting the intuitive insights into class parameters' behaviour.

A simple solution to handle the silver standard is to divide them evenly into bins. Then each bin is treated as a real class, converting continuous data to categorical. However, because of the nature of tweets and offensive languages, it is hard to define the offensiveness of tweets and offensive utterances are offensive in different levels. Some may have strong offensive signals but some are vague. Leveraging a dynamic system to learn these various signals is preferable.

More specific, for a classification task of k classes, we have $y_i \in \{0, 1, ..., K\}$ and corresponding class parameters $\sigma_y \in \mathbb{R}^K$, where the each entry is a learnable weight of the corresponding class. Let $z_i \in \mathbb{R}^K$ be the logits predicted by the network. To adapt the class parameters to semi-supervised data, we set $K = 10$ to divide the dataset into 10 bins. We therefore have $\sigma_y \in \mathbb{R}^{10}$, but the shape of outputted logits z_i remains unchanged. For the binary classification task, $z_i \in \mathbb{R}^2$. The logits z_i is scaled by the class parameters depending on the bin x_i in.

[Saxena et al., 2019] updates the data parameters σ_y using the negative of gradient. Thereby, the system will first focus on the easy samples, and then it turns its attention to hard samples gradually, which consumes numerous time and data. It is problematic to spend a large amount of time on training, particularly overfitting in a small dataset. Although it would be less impactful when it comes to a larger data, it is still preferable to designa light and fast-training system.

For consistency, we further fine tune our model based on the fine tuned model that we used in the last section. Since this model is already fine tuned on OLID for an epoch, we assume that it has been preliminarily adapted to the domain of offensive language and tweet language. Therefore the update rule for class parameters is modified. We update the class parameters using the exact value of the gradient, namely **inverse update**, which leads to totally opposite behaviour of class parameters. Although inversely operating the class parameters have been mentioned in [Saxena et al., 2019], they stop probing into the inverse update and update the class parameters regularly across all experiments. The impacts of inverse update will be illustrated in Figure 16 later on. The inverse update reduces the importance of easy cases during the whole training process, instead of up-weighting them before down-weighting (suggested in [Saxena et al., 2019]). Normally, it is common to optimize the parameters in order to minimize the overall loss causing by the data. However, to enforce the system focusing on the hard ones, it worth nothing to avoid the loss by tuning down the weights of the hard ones. On the contrary, they should be uplifted. In this way, our model will increase the attention to hard cases once the fine tuning starts. The underlying intuition is that, rather than mastering the knowledge already learnt, the system should focus on those have not, which accelerates the training by turning the attention to the hard samples earlier. Mathematically, in Eq. 11, the class parameters of well classified classes (with $\delta > 0$) decays, those $\delta < 0$ rising up. We conclude the two update strategies as follows:

- Inverse update: up-weighting the hard cases (misclassified ones and well-classified but lack of confidence) at the cost of increasing training loss.
- Regular update: up-weighting the easy cases (already well-classified ones) avoiding the increase of training loss.

Revealing the behaviour of class parameters We know that class parameters achieve better generalization on supervised dataset. In order to see how it works with semi-supervised data and show the ability to cope with noisy labels, we start from observing how the class parameters flow during training. The experiments are conducted on subsets of SOLID, where samples are annotated with averaged confidence scores ranging from 0 to 1. The higher score the more offensive it suggests. As described beforehand, we set 10 bins which divide the confidence score evenly. The

cardinality of the bins specify the difficulty level of the data included (e.g. bin 0.5 denotes the data scores in range $[0.5, 0.6)$.). Note that the difficulty is not monotonic. As the middle bins is more difficult than the bins at both sides. The class parameters $\sigma_{y_k}$for each bin is initiated as 1. We further fine tune the DistilBERT model for 2 epochs (learning rate=1e-5, batch size=64). We used stochastic gradient descent to update class parameters with learning rate of 1e-3, momentum of 0.9. Two figures showing the change of σ_y using different update rules are presented, which includes:

- updating σ_y through the negative of gradient;
- updating σ_y inversely (through the exact value of gradient).

Note that as it illustrated before, the increase of σ_y leads to down-weighting, in contrast, the decrease suggests up-weighting.

As shown in Figure 16, $\sigma_{y=0.4,0.5}$ keep decaying. It represents bin 0.4 and 0.5 are been up-weighted across the training, because these samples are usually showing veiled features for classification and therefore hard to classify. Even they are classified correctly, if we refer to Eq. 11, the δ for these classes are often low, which indicates the system is not confident enough to make that decision. Thus, the class parameters are increased, as a way to rise the attention to these hard samples. The least favourite bin is 0.2. Because samples within this bin rarely contain offensive terms or any veiled insults. The system performs so well in these data that it reduce the attention to learning on them. The curves of bin 0.0 and 0.9 are flattened all the time, of which reason is the nature distribution of scores. The extreme samples only account for a very small proportion of the training samples. Hence, the $\sigma_{y=0.0,0.9}$ are rarely updated. A totally opposite curve trend is shown in Figure 17, in which the difficult bins (bin 0.4 and 0.5) are down-weighted. However, the curve $\sigma_{y=0.5}$ shows a downward trend whilst the curves $\sigma_{y=0.2,0.3}$ show an upward trend. This phenomenon in due to the effect of the regularization term in which the system is penalized from greedily focusing on easy cases.

Experimenting class parameters with different training manners The experiments are designed with the purposes to observe

- the effects of employing class parameters on different volume of data;
- the potential to fit the semi-supervised data;

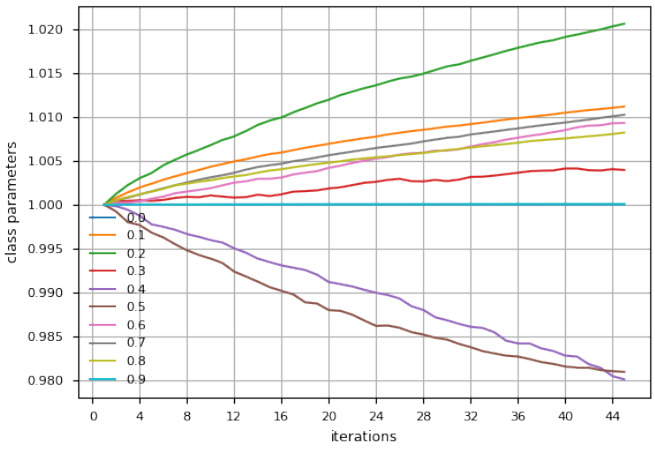

Figure 16: The values of class parameters changing with the training iteration. The x axis represents the training iteration and the y axis is the value of class parameters. 0.0 is used to denote the bin for data scoring $[0, 0.1)$, 0.1 to $[0.1, 0.2)$ and so on. The class parameters are updated inversely.

- how it integrate with different training manners (inverse update or not).

During inferencing, the confidence scores of the test samples are not accessible, the class parameters are no longer used. We conducted experiments shown on Table 14, in which Run-1, Run-4 and Run-7 are baseline models without using class parameters. To observe the performance on different data, we fine tune the models on different subsets of SOLID, which are SOLID-TH55, TH3558, and Th55f. The dataset notations are referred to Table 11 and Table 12. As suggested in the last section, the listed runs are further trained based on the model that is fine tuned on OLID. The highest results of different training datasets are in bold.

Among all the experiments above, under the same dataset, the models trained with class parameters outperforms the those not, even under the conditional subset TH3558, where extreme samples are eliminated. The models using inverse update outperforms those regular update. The reason may be that inverse update is suitable when the model is well-performed in the easy cases but requires further fine tuning in hard cases. This will be illustrated later in Table 15. As shown in Figure 17, it takes much longer iterations for regular updating class parameters to turning the model focus from the easy to the hard. With the prior knowledge of the dataset and class parameters, Run-6 (SOLID-TH3558) yields the best result in macro F1 among all the

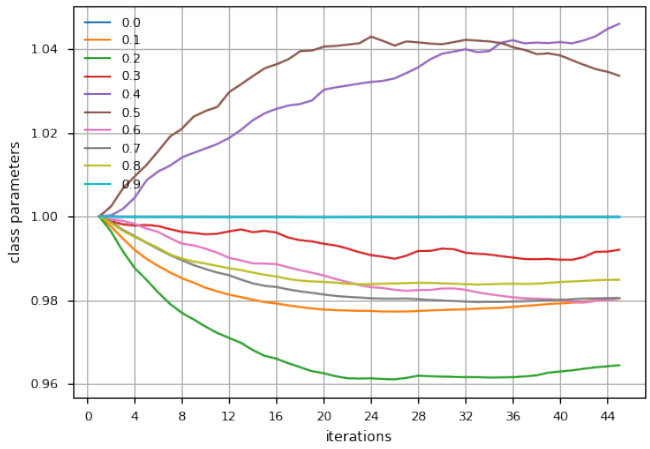

Figure 17: The values of class parameters changing with the training iteration. The class parameters are updated using the negative value of the gradient. The hyper-parameters and notation stick with Figure 16

experiments and is competitive in terms of F1-1 with the run-9 who is fine tuned on the over 20 time larger dataset and yield the best in F1-1.

We find that with class parameters, involving extreme samples are not as harmful as we showed in Table 11. With a small proportion of down-weighted extreme samples, Run-3 outperforms Run-4 who is fine tuned on the conditional subset excluding extreme samples. Note that both of Run-4 are fine tuned with the conditional subset SOLID-TH3558 who shows the significant improvement in the continual selective learning experiments (Table 12). Based on the previous effort (SOLID-TH3558), Run-5 and Run-6 both surpass the performance of Run-4. Among these experiments, the overall performance on small-scaled dataset such as SOLID-TH55 and SOLID-TH3558 are competitive, (for SOLID-TH3558, even superior) to the performance on the larger size data (SOLID-TH55f). This shows the capability of applying class parameters to small size dataset.

In conclusion, the class parameters shows the capability to confront with the noisy labels with small data. Because of the continuous down-weighting of extreme samples in which most of them are well-classified, the negative impacts are smoothed out. Besides, the class parameters enable the system to dynamically decide how much the system learns from the training samples.

It is interesting to note that, it usually takes more epoch for class parameters

Runs	Further fine tune on		F1 (%)	Class Parameters	Inverse Update
Run-1	SOLID-TH55	Macro F1	.788	False	False
		F1-1	.718		
Run-2		Macro F1	.793	True	False
		F1-1	.717		
Run-3		Macro F1	**.795**	True	Ture
		F1-1	**.719**		
Run-4	SOLID-TH3558	Macro F1	.794	False	False
		F1-1	.723		
Run-5		Macro F1	.796	True	False
		F1-1	.725		
Run-6		Macro F1	**.803**	True	True
		F1-1	**.733**		
Run-7	SOLID-TH55f	Macro F1	.799	False	False
		F1-1	.720		
Run-8		Macro F1	.796	True	False
		F1-1	.715		
Run-9		Macro F1	**.801**	True	True
		F1-1	**.737**		

Table 14: Results of applying class parameters to DistilBERT models. Evaluation is done over the OLID development set. The stochastic gradient descent is applied to update class parameters (learning rate of 1e-3 and momentum of 0.9). The relevant models are trained for 3 epochs on SOLID-TH55 and TH3558 with learning rate 1e-5, batch size 64, and 1 epoch on SOLID-TH55f.

to show its potential. During the experiments, training for 3 epochs on a small size subset leads to overfitting in most our models. However, training a model with class parameters requires more epochs.

Further, how the class parameters influence the model inference is also investigated. The training configurations in Table 14 are tested on a silver standard validation set, from which we can eyeball how the model performs with different levels of test data. Specifically, the validation set is divided into 10 bins by the scores and assume the samples scoring above 0.5 as label OFF and the rest as NOT. Therefore, the results are away from calibrate and can not be viewed as model evaluation, however, it can provide us some insight on the model attention and the underlying motivations of leveraging dynamic CL. Each data bin is used for model validation respectively and their accuracies score are provided. The results are shown in Table 15.

As shown in Table 15, the classification accuracy are almost flawless among the bins below 0.2 and above 0.7, while the bins in the middle are error prone. Comparing the runs without class parameters (Run-1,4, and 7) with those applying class parameters

Runs	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Run-1	1.0	0.998	0.998	0.952	0.769	0.679	0.927	1.0	0.998	1.0
Run-2	1.0	0.999	0.998	0.961	0.773	**0.684**	0.934	1.0	0.998	1.0
Run-3	1.0	0.999	0.998	0.965	**0.792**	0.675	0.923	1.0	0.998	1.0
Run-4	1.0	0.999	0.994	0.942	0.686	0.77	0.952	1.0	1.0	1.0
Run-5	1.0	1.0	0.998	0.931	0.667	**0.789**	0.959	1.0	1.0	1.0
Run-6	1.0	1.0	0.999	0.95	**0.725**	0.733	0.945	1.0	1.0	1.0
Run-7	1.0	1.0	0.999	0.989	0.824	0.744	0.97	1.0	1.0	1.0
Run-8	1.0	1.0	1.0	0.994	**0.898**	0.615	0.959	1.0	1.0	1.0
Run-9	1.0	1.0	0.999	0.933	0.596	**0.899**	0.984	1.0	1.0	1.0

Table 15: Accuracy on SOLID validation set. The header indicates the difficulty level of data bins. Note that the difficulty is not monotonic. As the middle bins is more difficult than the bins at both sides. The 9 runs correspond to the training configurations from Table 14. They are accordingly divided into 3 sections by the different training data they use. All runs perform almost perfect except in the bin 0.4 and 0.5. thereby, the highest results of different training datasets in bin 0.4 and 0.5 are in bold.

under the same section, the accuracy of middle bins are increased, with the accuracy of well-classified bins almost unchanged. These numbers embody the general idea of the class parameters who applies a dynamic curriculum over learning the samples with multiple difficulties. Focusing on the hard cases is nearly intact to the performance of the easy ones, resulting in a general and robust model.

Over all the above experiments, inversely update the class parameters using stochastic gradient descent (learning rate 1e-3 and momentum 0.9) in the SOLID-TH3558 subset yields the best result. The above experiments demonstrate that with the binning effects in the class parameters, the models gain the ability to confront the corrupted silver standards, which interest us to further investigate the automatic weighting of all difficulty levels via loss functions. Rather than using the fixed size class parameters, loss functions offer flexibility on controlling the learning pace for each sample with its own difficulty.

3.5 Experiment with Dynamic Loss Functions

Before stepping into dynamic loss functions, it is necessary to introduce a parametrizable loss function. We have seen many studies modified the standard cross entropy

(CE) loss such as Symmetric CE loss (SL) [Wang et al., 2019] and focal loss (FL) [Lin et al., 2017]. They both embed task specific empirical evidence to scale the CE loss and yield promising results. In terms of the dynamic loss functions, [Wu et al., 2018] proposed the gradient based, teacher-student framework to optimize the loss function and achieved progress in neural machine translation (NMT) and image classification. However, we see nascent studies leveraging scaled loss functions or dynamic approach to attain the optimized loss function in the field of NLP or any sentiment related classification tasks. Further in our task, fitting a massive volume of semi-supervised data has been a tough challenge. To the best of our knowledge, most of the practices were done over supervised learning, few with semi-supervised learning. Moreover, the nature of offensive language and the large volume noisy data even complicate the problem. Many researchers who have taken participate in the SE19T6A and SE20T12A would rather use the smaller but labeled data. Although the resulting systems yield good performance, they may suffer from data bias and lack of generalization capability because of the content of OLID. [Wang et al., 2019] pointed out that learning with standard CE loss may cause hard class under-fitting problem, which also give rights to exploit dynamic loss functions in this field.

We see the dynamic loss functions' potential to adapt to the semi-supervised data. Taking the challenge, the scaled loss functions and dynamic optimization framework are therefore leveraged on the offensive language identification task. We first illustrate the intuition of designing our scaled loss function and how the optimization framework is applied to the scaled loss function. Afterward, corresponding experiment results are presented.

To explain our scaled loss functions, we define **system difficulty** as the gap between the system's predicted probability and expected probability for gold standard label; and **sample difficulty** as the objective difficulty to classify flawlessly. The modulating factor can be seen as the measurement of system difficulty. As its value ascending, the gap between prediction and gold standard being wider, it is hard for the system to well-classified the sample, and vice versa.

One of the limitations of FL is that it is not designed for continuous labels. Although it can be addressed by binning, converting continuous labels to discrete, the continuous information is flatten out, which causes chaos in the system. Another way to deal with continuous data is treating it as a regression task. However, we find it

very difficult fitting the data and underperforming. Therefore, in order to adapt the semi-supervised data, we employ the idea (strengthening CE loss with scaling factors), which is shown to be interact effectively with our data. Inspired by FL, in order to make our scaling factor work like weights, a Gaussian function are used as a scaling function that reshapes the CE loss is employed in the experiments.

$$g(q_t) \;=\; \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{q_t-\mu}{\sigma}\right)^2} \tag{21}$$

where μ and σ represent the mean and standard deviation, which jointly control the shape and central of the scaling function. A more intuitive illustration is provided in figure 18. Then, The Min-max normalization is used to normalize the Gaussian function:

$$g\prime(q_t) = \frac{g(q_t)}{g(\mu)} \tag{22}$$

Similar to p_t, $q_t \in [0.5, 1]$ is defined as:

$$q_t = \begin{cases} y^* & y^* > 0.5 \\ 1 - y^* & \text{otherwise} \end{cases} \tag{23}$$

where y^* is the silver standard score, ranging from 0 to 1. The Equation above maps the non-offensive score from $[0, 0.5)$ to $[0.5, 1)$ for the sake of unity. In Figure 18, we demonstrate the $g'(q_t)$ with different μ and σ, in which the μ controls the central position which is the top focus region and the σ controls the shape. The steeper the shape is, the lesser the focus shared other than the central. $g'(q_t)$ can be viewed as a modulating factor as $(1 - p_t)^\gamma$ in FL. It is then used for scaling the standard CE loss:

$$FL'(p_t, q_t) = -g'(q_t) log(p_t) \tag{24}$$

Similar to p_t, q_t can be seen as the indicator of sample difficulty, which describes the objective difficulty annotated via silver standard scores. The extreme cases are considered as easy cases since they are obvious for human, because they carry strong signal of offensiveness. However, the middling cases are vague and ambiguous in semantics, therefore these are hard to classify correctly.

To combine the two concepts, the system difficulty and the sample difficulty are integrated into one unified loss function, namely **Dual focal loss** (Dual FL). Dual

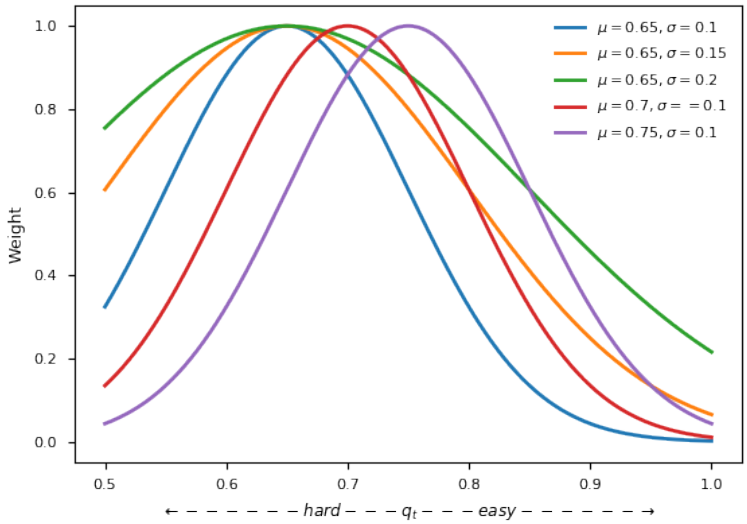

Figure 18: Plotting $g'(q_t)$ with different μ and σ

FL, as its core, is a scaled cross entropy loss. It takes the sample categorical label and estimated difficulty as input, outputs the scaled CE loss, which is feasible for fitting semi-supervised data. The dual focal loss are given by equation 25.

$$\text{Dual FL}(p_t, q_t, \tilde{y}) = -p_t * g'(q_t) log(p_t) \tag{25}$$

where $\tilde{y}$ is the categorical label. Normally, in most sentiment related task, 0.5 is the threshold to divide positive and negative sentiment. This integrated loss function possesses the property of down-weighting the well-classified samples as well as adjusting the weight depending on its difficulty. The cross influence of sample difficultness and system difficultness is illustrated in Table 16, in which the arrow indicates whether the loss is up-weighted or down-weighted by Dual focal loss.

system difficultness \ sample difficultness	Hard	Easy
Hard	↑	↑↓
Easy	↑↓	↓

Table 16: Cross influence of sample difficulty and system difficulty. The dual impacts come from q_t and p_t respectively. When the difficulty indicators are divergent, the impact is deducted by each other. The exact direction is determined by the system.

Shown in Table 16, what differs Dual FL from FL is the ability to rectify the loss

through its sample difficulty. The system will still get some bonus from correctly classifying high sample difficulty samples. What FL brings in is the miss-classifying a low sample difficulty sample is penalized. To investigate how Dual FL loss alleviates the distraction of noisy labels, we run an ablation study on the parameters of Dual FL loss. Specifically, based on the same model we used for previous experiments, we further fine tune the DistilBERT model on SOLID-TH55 for one epoch with learning rate 1e-5. We experiment the μ in $0.5, 0.6, 0.65, 0.7, 0.8, 0.9$ and σ from 0.1 to 1.2 with a step of 0.03, plus $\sigma = 0.12$, $6 * 38 = 228$ runs in total. For comparison, we experiment standard FL with the default hyper parameters in [Lin et al., 2017], shown as a constant value in Figure 19. In the figure, despite some poor performance in a few extreme steep function shapes, most of the results fall around 0.788. From Figure 18 we know that μ controls the central focus of Dual FL and its shape is changed by σ. In general, $\mu = 0.65$ yields the most promising results, the following is the curve $\mu = 0.7$. This suggests that focusing on the medium hard samples is beneficial. The trending of curve $\mu = 0.5$ indicates putting excessive focus on the hard samples could have counter effects. The curve gradually increases as its shape becomes less steep and the focus is dispersed. Comparing to others, curve $\mu = 0.8, 0.9$ does not severely underperforming when providing suitable hyper-parameters, ditto to $\mu = 0.5$. This is because focusing on the minor extreme easy samples has trivial impact on the system. But the performances are unacceptable once without hyper-parameter tuning. The curve $\mu = 0.65$ peaks at both $\sigma = 0.12$ and $\sigma = 1.0$ outperforming the standard FL by 1.5%. This shows the potential of unevenly scatter the focus over different difficulties using scaled loss functions. With the small modification to standard FL, the dual FL adapts better to the semi-supervised data given the proper hyper-parameters. In this case, by setting $\mu = 0.65$ and $\sigma = \{0.12, 1.0\}$, the scaled loss outperforms the standard FL by 1.5%. The small invisibility of improvement is due to the simplicity of the test data. As shown in the baseline study, many baseline approaches are able to produce promising results such that a small improvement requires considerable efforts. However, to obtain the best performance, the ablation study must be performed (ditto to the FL), which accounts for huge overhead to search through the parameter space and gain the heuristic of the best pick. We are not claiming the unworthiness of ablation study but the costly overhead need to be automated. It is necessary to perform an ablation to obtain the optimal hyper-parameters. But considering that

the huge overhead would be taken, designing dynamic loss functions to eliminate the manual parameter tuning is prioritized as an option at a lower cost.

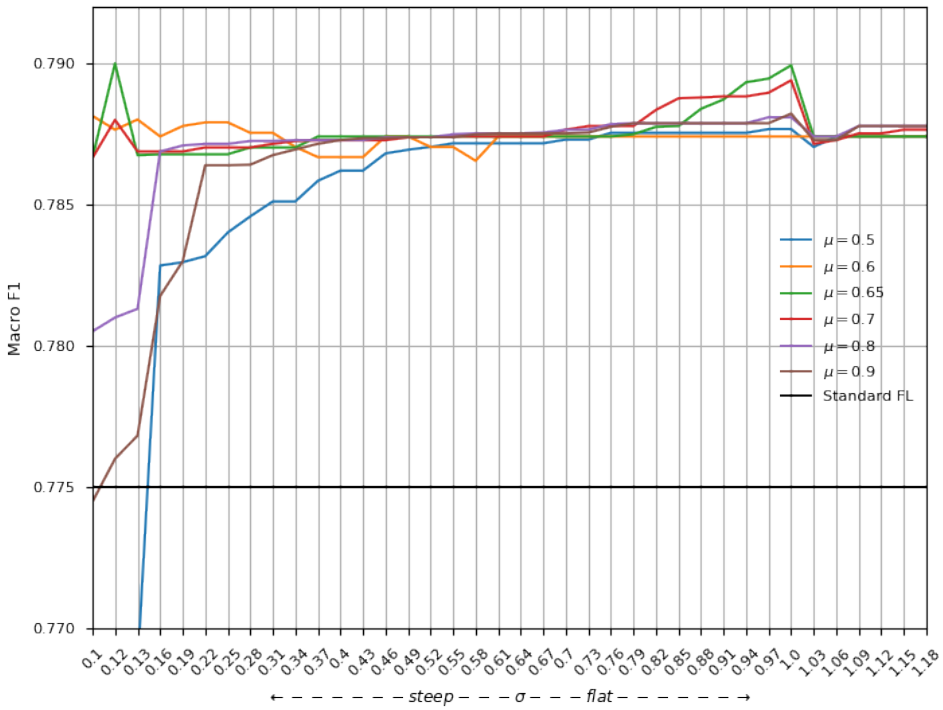

Figure 19: Ablation study for different μ and σ combinations in scaled loss functions

For the rest of the section, a dynamic curriculum is applied to the scaled loss functions. Considering the costly time consuming issue in replicating the gradient based, teacher-student framework, we simplify the learning framework such that it can be applied on the huge amount of semi-supervised data with less time consuming. To this end, we leverage the stand-alone SOLID development set to guide the teacher model. The SOLID development set is split from the SOLID, containing 5000 silver standard samples. In contrast to [Wu et al., 2018] who trains the teacher model after the student model, we modify the training logic so that the teacher model is trained along with the student model. Our training procedure is described as follow.

1. Initialize the teacher model g_θ and load the student model f_w. w and θ are learnable model parameters.

2. Evaluate the student model f_{w_0} in SOLID development set and get the current state vector s_t.

3. If the current time step is 0, Input the s_0 to the teacher model and output loss parameters μ_0 and σ_0.

4. Fine tune the student model with training data for I iterations and update the current w to w_t.

5. Redo second step with f_{w_t} and update the teacher model.

6. Go to step four and continue the fine tuning with s_t and new parameters μ_t, σ_t.

I is an adjustable parameter controlling the frequency of updating the teacher model. The higher I is, the teacher model is lesser often updated. Similar as [Wu et al., 2018], the teacher model outputs the hyper parameter of the student model (the model for our task), in our case, the μ and σ in the scaled loss function. The evaluation manner in SOLID development set is identical to Table 15, which results in 10 accuracy scores for 10 data bins. The accuracy vector is used for the state vector. With the dynamic optimization framework, it is unnecessary to manually initialize the μ and σ. They are randomly initialized from $\mu \in [0.5, 1)$ and $\sigma \in (0, 2)$. To reduce the model complexity, the teacher model is a one layer feed-forward network with 10 hidden neurons. Addressing the hard case under-fitting issue, accuracy vector is used as features as input to teacher model. Besides, since the teacher model is trained on the SOLID development data, it accordingly optimizes the outputted parameters to adapt to the different difficulties. The advance of the teacher-student model lies on the ability to take into account of the state of the student and output appropriate parameters at current state.

To update the teacher model efficiently, the update approach in [Wu et al., 2018] is simplified. The RMD based teacher model update is ditched and we simplify update rule of the teacher model as follows. The loss outputted by the teacher is written as $L_{g_\theta(s_t)}$. The teacher model is trained via the gradient of loss w.r.t its parameters θ, that is $\sum_{x,y \in D_{\text{dev}}} \frac{\partial L_{g_\theta(s_t)}(f_{w_t}(x), y)}{\partial \theta}$, where $x, y \in D_{\text{dev}}$ denotes the training samples in SOLID development set. With the gradient, the θ is updated via SGD with learning rate 2e-5 in our experiment.

In the experiments, the DistilBERT is used as student model, outputting the prediction and the one layer feed-forward network as teacher. The dynamic scaled loss functions are used as the criterion. Figure 20 shows the how the accuracy of each difficulty evolves during the fine tuning and Figure 21 is an example of how the exact parameters μ and σ change. As denoted before, colors distinguish the data bins of different difficulties. At the beginning stage, bin 0.3 and 0.4 are relatively under-performing whilst the other bins are at high accuracy level (above 85%). Since the system difficulty is introduced before, the difficulty is never static, but changes as model being fine tuned. In this case, bin 0.3 and 0.4 are the difficult bins at that moment. During the fine tuning, the performances for all bins fluctuate with the under-performing bins gradually rise up. Although the bin 0.5 falls from accuracy of 90%, it recovers after iterations and eventually approaches to the initial performance. Bin 0.3 and 0.4 increase largely by nearly 20% compared to their initial accuracy. For those already well-performed bins, they are less volatile than those difficult bins and continue showing the high accuracies. The accuracies flowing in Figure 20 demonstrates that the teacher-student optimization frame we applied to the loss functions is able to take the account of the current state of the model and adjust the parameters as a way to scatter the focus properly over the under-performing cases and the well-classified cases. The performance of easy cases is almost intact and the poor performing cases are improved. However, due to the corrupted labels in SOLID, the performances of difficult bins fail to end up with high accuracies but still fluctuate within a small range. This showcases that the poor annotation quality may distract the teacher model and the obstacle of fitting in such corrupted silver standards.

Table 17 reports the results on the OLID development set. For comparison, the results of previous experiments are pasted. For comparison Run 1, 4 and 7 are models without dynamic loss. These runs are fine tuned on different datasets: SOLID-TH55, SOLID-TH3558 and SOLID-TH55f for 1 epoch. The properties of these datasets are referred to Table 11 and Table 12. The models are evaluated over the OLID development set. Using the same policy, the models are further fine tuned based on a tuned base model. The teacher model are trained with learning rate 2e-5 and stochastic gradient descent (SGD) is used for optimization. The training interval I is set to 19, which means the teacher model is updated every 19 iterations of the student model.

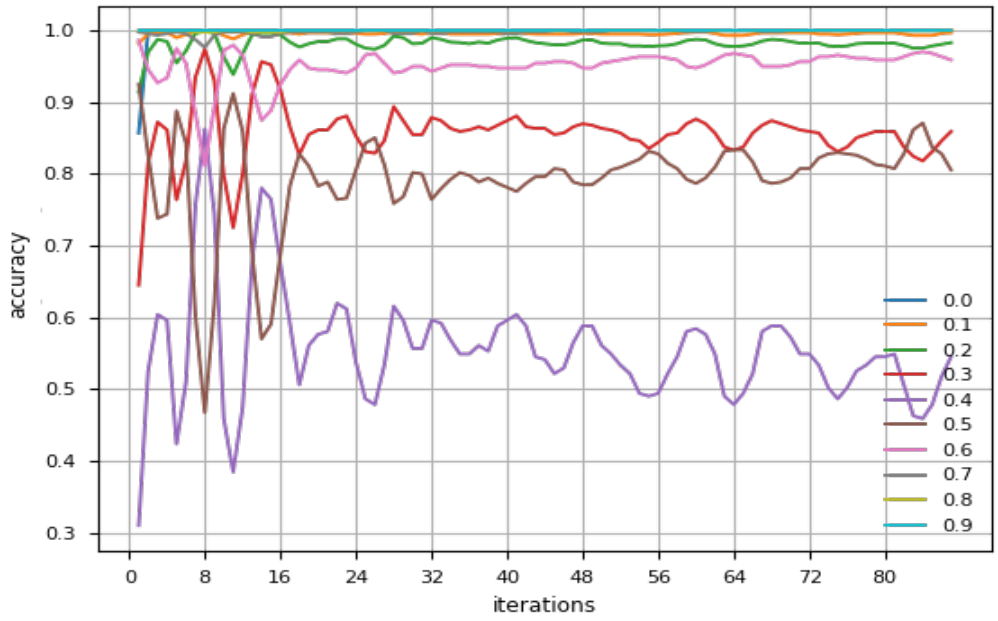

Figure 20: The accuracy of each data bin. In general, the middling bins are considered difficult away from which are those relatively easy bins.

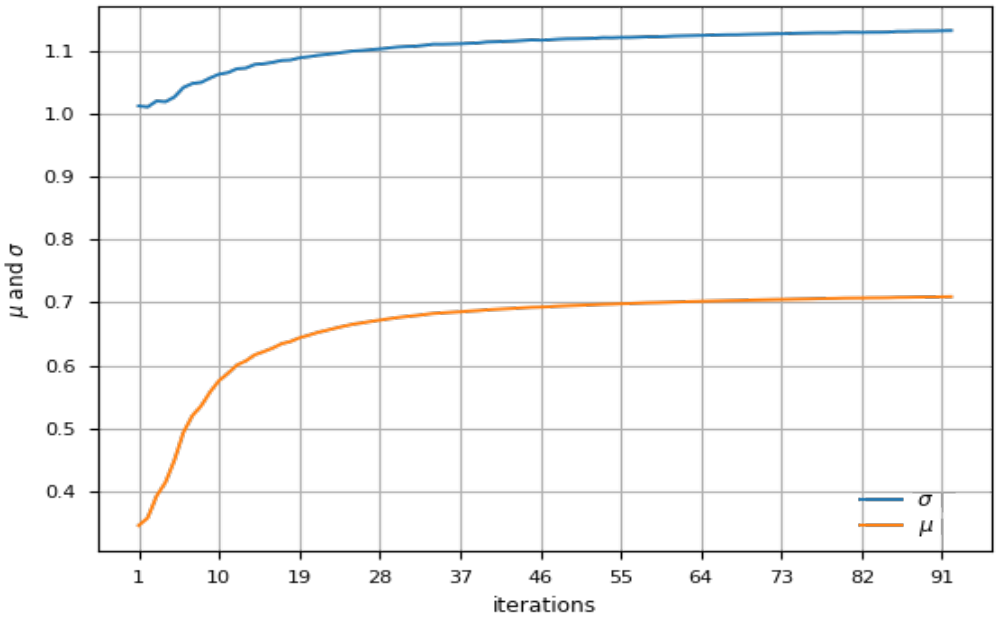

Figure 21: The values of μ and σ

Runs	Further fine tune on		F1 (%)	Dynamic loss
Run-1	SOLID-TH55	Macro F1	.788	False
		F1-1	.718	
Run-10		Macro F1	.793	True
		F1-1	.719	
Run-4	SOLID-TH3558	Macro F1	.794	False
		F1-1	.723	
Run-11		Macro F1	.796	True
		F1-1	.724	
Run-7	SOLID-TH55f	Macro F1	.799	False
		F1-1	.720	
Run-12		Macro F1	.795	True
		F1-1	.722	

Table 17: The results of using dynamic loss functions on OLID development set.

Comparing with the the results of class parameters in Table 14, The dynamic loss function approach is slightly under-performing but still yields better performances compared to the baselines, the standard FL, and the scaled loss functions. on the small-scaled data (SOLID-TH55 and SOLID-TH3558). The reason on the limited increase is speculated that the structure of the teacher model is relatively plain, which may not adapt to the student model perfectly. This may incur overfitting of the teacher model. Moreover, given that the teacher is trained on SOLID development set which is a semi-supervised dataset and mislabeling of samples are inevitable, the teacher may lose the calibrate estimation of the current proper parameters. Besides, the state vector contains the accuracies of data bins in which most of the accuracies are stable with only a few volatile data bins for hard cases. Therefore, the state vector may not consist enough informative features.

Despite these drawbacks, dynamic loss yields better performance under the small size of data and requires less manual hyper-parameters tuning. To obtain the relatively optimized hyper-parameters, it is common to perform an ablation study on all the hyper-parameters or a random search of parameter space. However, it is not necessarily precise and causes huge training overhead. The dynamic loss largely reduces the overhead by automating the precess and offers flexibility upon the hyper-parameters with the better adaptation to the noisy silver standards than the standard FL as well

as scaled loss.

3.6 Performance on Official Test Data

This section evaluates the class parameter models and the dynamic loss function models on OLID official test set and SOLID official test set respectively. Shown in Table 18, the best results for OLID and SOLID test data are in bold. For comparison, a baseline of the DistilBERT model and the SOTA results in SE19T6A and SE20T12A are provided.

Run		OLID	SOLID	Experiment type
DistilBERT	Macro F1	.807	.909	-
	F1-1	.713	.875	
SOTA-OLID	Macro F1	.829	-	-
SOTA-SOLID	Macro F1	-	.920	
Run-2	Macro F1	.816	.910	Class parameters
	F1-1	.726	.876	
Run-3	Macro F1	.816	.910	
	F1-1	.725	.876	
Run-5	Macro F1	.818	.910	
	F1-1	.732	.876	
Run-6	Macro F1	**.819**	.908	
	F1-1	**.734**	.876	
Run-8	Macro F1	.814	**.912**	
	F1-1	.719	**.879**	
Run-9	Macro F1	.798	.910	
	F1-1	.709	.877	
Run-10	Macro F1	.813	.907	Dynamic loss
	F1-1	.719	.875	
Run-11	Macro F1	.815	.908	
	F1-1	.723	.875	
Run-12	Macro F1	.816	.910	
	F1-1	.724	.877	

Table 18: Results on OLID official test set and SOLID official test set.

The Run-6 is selected as the best run because it performs the best in OLID which

is believed as a harder task compared to SOLID. Table 19 lists some real test examples that are misclassified by the baseline DistilBERT model but are well-classified in the best runs (Run-6 and Run-8) and otherwise.

The first tweet is labeled as OFF because of the superlative adjective *the most incompetent*. The second and third are lengthy and the second one is being offensive implicitly. The offensive trigger of the forth tweet *shyt* has spelling error and is correctly classified. The last three tweets of first section discussing the crime and murder are labeled as NOT. The are well-classified even they carry strong offensive signal such as *getting murdered*, *commit murder*, and *thugs*. These examples may be considered as hard in the sense of text length, semantics, figurative language, metaphor etc.. This suggests that focusing on the hard samples during fine tuning improves the performance on some tricky samples.

The second section demonstrates the the examples that are correctly predicted by the baseline DistilBERT model but are misclassified in the best model. The first two examples contain implicit offensive signal such as the metaphor *parasite glazers* and *humiliate*. Especially the second tweet which is veiled is offensive when considering the context semantics. The rest of the examples are controversial. The third examples are not offensive although it contains a profanity *damn*. It is totally acceptable and normal to post such a tweet and means no offence. Such error is inevitable because of the controversy itself and mitigating this sort of problem requires further work. The last three examples are more offensive than their labels where they either expresses hate towards gays, muslims or show hostility to a person. However, it could not simply blame to the annotation error. Annotator may hold different interpretations to controversial examples. Thus, it is acceptable for the model to misclassify these examples. Without progresses in natural language reasoning and truly understanding the hidden semantics of machine, these tricky samples may not be truly solved. In addition, because of the silver nature in the SOLID test data, annotation errors are inevitable. The last three samples could be the examples of mislabeling samples.

Dataset	Gold standard	Silver standard	System prediction	Tweet
OLID	OFF	-	OFF	*#Nigeria #Naija #9ja 'You are the most incompetent leader in Nigeria's history' – Atiku blasts Buhari URL By championnewsng Via URL*
OLID	OFF	-	OFF	*#BoycottNike campaign must continue and grow Paying overseas staff $0.20 an hour is not okay because your staff is over there". You are exploiting poor women/men in the world. This is more than Americans can stand. URL*
OLID	OFF	-	OFF	*#Antifa: Take note of how to protest with civility. This is why socialism (aka communism) will never win. It is inherently evil and unsustainable. URL*
SOLID	OFF	-	OFF	*@USER Trump is a piece of shyt human being*
OLID	NOT	-	NOT	*@USER @USER @USER @USER @USER @USER @USER @USER @USER @USER Gun control doesn't stop kids and people getting murdered by guns. Chicago has had gun control for decades and it has multiple shootings every week.*
OLID	NOT	-	NOT	*@USER #MAGA who cares about the farm. He had no reason to commit murder. End of story.*
OLID	NOT	-	NOT	*@USER Be sure to send out the left's antifa thugs.*
SOLID	OFF	-	NOT	*@USER Get the parasite glazers out of our club. Enough is enough #glazersout*
OLID	OFF	-	NOT	*@USER Do you get the feeling he is kissing @USER behind so he can humiliate him later?*
SOLID	OFF	-	NOT	*Sometimes when I look into your eyes,I pretend you're mine, all the damn time.*
SOLID	-	NOT	OFF	*@USER Gay people Are cause by to be able to detect their own fault*
SOLID	-	NOT	OFF	*@USER Muslims pushing the own agenda that's all that is, playing victim,* 🤮🤮🤮
SOLID	-	NOT	OFF	*what border agent let those snakes on a plane? did they have passports? i bet you those snakes did not have any legal identification*

Table 19: Error examples from official test data of OLID and SOLID. The first section shows the examples that are misclassified by the baseline DistilBERT model but are well-classified in the best run while the second section lists the otherwise. Note that SOLID holds gold standard for OFF labels but silver standard for NOT labels.

Chapter 4

Conclusion and Future Work

This thesis investigates the curriculum learning (CL) and dynamic loss functions in offensive language identification. Confronting the scarce human annotated data, the potential of leveraging the semi-supervised data by means of CL and dynamic loss functions is tentatively explored. SemEval datasets OLID and SOLID are compared from the perspective of their lexical properties and from the difference in their annotation standards (gold and silver standard), as well as the content and phrasing, which all are different.

The main goal of the thesis is not to attempt to beat SOTA on the datasets used. Rather, this thesis explores ways to reduce the complexity of the processing to address a very large dataset as automatically as possible without prohibitive resource allocation. Regarding the costly training overhead, the DistilBERT model is selected because it is relatively lite and is able to achieve competitive results with less parameter size and training data, compared to other heavy pre-trained models such as BERT and RoBERTa.

The first idea explored is forms of downsampling. The preliminary experiments on selective further fine tuning with SOLID reveal that by excluding the extreme samples (easy cases), the model gains improvement beyond simply balancing the classes.

This insight leads to the automatic curriculum learning based on the difficulty of samples. In particular, the hard cases are up-weighted while the easy cases are down-weighted. This idea is implemented via class parameters which assign learnable weights to different difficulty levels.

To see if selective loss functions could offer more flexibility on the control of

difficulty, this thesis further investigates the idea of dynamic loss. To this end, a new loss function Dual Focal Loss (DFL) is parameterized based the standard Focal Loss. DFL takes into account the system difficulty (the gap between system prediction to expected prediction) and sample difficulty (the objective difficulty indicating via the annotation). A teacher-student framework is applied to optimize the DFL dynamically, which allows the model adapt the dynamic attention to different difficulty levels of data as well as adapting better to SOLID.

Both the class parameter approach and the dynamic loss show better results on the silver standard and are robust when confronting the hard cases without impacting the performance on easy cases.

The performance of our best model is approaching the SOTA system on SE19T6A and SE20T12A tasks. Combining curriculum learning with dynamic loss functions, the model lags SOTA by only 1% F1 score on SE19T6A and 1.2% on SE20T12A with at least 50% parameter size reduction and less data overhead, compared to the SOTAs on each task.

Given the increasing use of large silver standards in the field of NLP, this thesis contributes approaches that mitigate the effects of questionable annotations on highly imbalanced, huge datasets.

As for the future work, what is insufficient in evaluating the performance of offensive language detection models is that: with the F1 score, one may preliminary perceive the overall performance, but lack of the insight of concrete performances with respect to samples of different difficulties. Given that the easy cases account for the majority of the offensive language on social media, simple architectures may yield acceptable performance. Building such versatile systems requires delicate efforts in difficulty evaluation. We notice that before fine tuning a pre-trained model, some studies further train the language model with the masked language model (MLM) task, which shows promising results. It worth further investigation on whether the CL or dynamic loss can accelerate or strengthen the further training. In addition, it is also worthwhile to further investigate the other forms of verbal cyber-bullying with CL and dynamic loss.

Bibliography

[Barbieri et al., 2020] Barbieri, F., Camacho-Collados, J., Espinosa Anke, L., and Neves, L. (2020). TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online. Association for Computational Linguistics.

[Bengio et al., 2009] Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

[Biesialska et al., 2020] Biesialska, M., Biesialska, K., and Costa-jussà, M. R. (2020). Continual lifelong learning in natural language processing: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6523–6541, Barcelona, Spain (Online). International Committee on Computational Linguistics.

[Burnap and Williams, 2015] Burnap, P. and Williams, M. L. (2015). Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242.

[Caselli et al., 2021] Caselli, T., Basile, V., Mitrović, J., and Granitzer, M. (2021). Hatebert: Retraining bert for abusive language detection in english.

[Cirik et al., 2016] Cirik, V., Hovy, E., and Morency, L.-P. (2016). Visualizing and understanding curriculum learning for long short-term memory networks.

[Conneau et al., 2020] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. In *Proceedings of*

the 58th Annual Meeting of the Association for Computational Linguistics, pages 8440–8451, Online. Association for Computational Linguistics.

[Cunningham et al., 2002] Cunningham, H., Maynard, D., Bontcheva, K., and Tablan, V. (2002). Gate: A framework and graphical development environment for robust nlp tools and applications.

[Dadu and Pant, 2020] Dadu, T. and Pant, K. (2020). Team rouges at SemEval-2020 task 12: Cross-lingual inductive transfer to detect offensive language. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2183–2189, Barcelona (online). International Committee for Computational Linguistics.

[Dai et al., 2020] Dai, W., Yu, T., Liu, Z., and Fung, P. (2020). Kungfupanda at semeval-2020 task 12: Bert-based multi-task learning for offensive language detection.

[Dauphin et al., 2012] Dauphin, G. M. Y., Glorot, X., Rifai, S., Bengio, Y., Goodfellow, I., Lavoie, E., Muller, X., Desjardins, G., Warde-Farley, D., Vincent, P., Courville, A., and Bergstra, J. (2012). Unsupervised and transfer learning challenge: a deep learning approach. In Guyon, I., Dror, G., Lemaire, V., Taylor, G., and Silver, D., editors, *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pages 97–110, Bellevue, Washington, USA. PMLR.

[Davidson et al., 2017] Davidson, T., Warmsley, D., Macy, M., and Weber, I. (2017). Automated hate speech detection and the problem of offensive language. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 11.

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

[Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

[Elman, 1993] Elman, J. L. (1993). Learning and development in neural networks: the importance of starting small. *Cognition*, 48(1):71–99.

[Festinger et al., 1963] Festinger, L., Pepitone, A., and Newcomb, T. M. (1963). Some consequences of de-individuation in a group.

[Ghosh et al., 2017] Ghosh, A., Kumar, H., and Sastry, P. S. (2017). Robust loss functions under label noise for deep neural networks.

[Gitari et al., 2015] Gitari, N. D., Zuping, Z., Damien, H., and Long, J. (2015). A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4):215–230.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

[Greco et al., 2019] Greco, C., Plank, B., Fernández, R., and Bernardi, R. (2019). Psycholinguistics meets continual learning: Measuring catastrophic forgetting in visual question answering.

[Harris, 1954] Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

[Hinton et al., 2015] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[Hong et al., 2018] Hong, X., Wong, P., Liu, D., Guan, S.-U., Man, K. L., and Huang, X. (2018). Lifelong machine learning: Outlook and direction. In *Proceedings of the 2nd International Conference on Big Data Research*, ICBDR 2018, page 76–79, New York, NY, USA. Association for Computing Machinery.

[Jiang et al., 2018] Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. (2018). MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2304–2313. PMLR.

[Joulin et al., 2017] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

[Kemker et al., 2017] Kemker, R., McClure, M., Abitino, A., Hayes, T., and Kanan, C. (2017). Measuring catastrophic forgetting in neural networks.

[Kim, 2014] Kim, Y. (2014). Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kocmi and Bojar, 2017] Kocmi, T. and Bojar, O. (2017). Curriculum learning and minibatch bucketing in neural machine translation. *RANLP 2017 - Recent Advances in Natural Language Processing Meet Deep Learning*.

[Kumar et al., 2010] Kumar, M., Packer, B., and Koller, D. (2010). Self-paced learning for latent variable models. In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc.

[Kumar et al., 2018] Kumar, R., Ojha, A. K., Malmasi, S., and Zampieri, M. (2018). Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

[Laaksonen et al., 2020] Laaksonen, S.-M., Haapoja, J., Kinnunen, T., Nelimarkka, M., and Pöyhtäri, R. (2020). The datafication of hate: Expectations and challenges in automated hate speech monitoring. *Frontiers in Big Data*, 3:3.

[Li and Hoiem, 2016] Li, Z. and Hoiem, D. (2016). Learning without forgetting. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision - 14th European Conference, ECCV 2016, Proceedings*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in

Bioinformatics), pages 614–629, Germany. Springer. Funding Information: This work is supported in part by NSF Awards 14-46765, 10-53768 and ONR MURIN000014-16-1-2007. Publisher Copyright: © Springer International Publishing AG 2016.; 14th European Conference on Computer Vision, ECCV 2016 ; Conference date: 11-10-2016 Through 14-10-2016.

[Lin et al., 2017] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007.

[Liu et al., 2019a] Liu, P., Li, W., and Zou, L. (2019a). NULI at SemEval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 87–91, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

[Liu et al., 2019b] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019b). Roberta: A robustly optimized bert pretraining approach. cite arxiv:1907.11692.

[Ma et al., 2020] Ma, X., Huang, H., Wang, Y., Romano, S., Erfani, S., and Bailey, J. (2020). Normalized loss functions for deep learning with noisy labels. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6543–6553. PMLR.

[Mahata et al., 2019] Mahata, D., Zhang, H., Uppal, K., Kumar, Y., Shah, R. R., Shahid, S., Mehnaz, L., and Anand, S. (2019). MIDAS at SemEval-2019 task 6: Identifying offensive posts and targeted offense from Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 683–690, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

[Malmasi and Zampieri, 2017] Malmasi, S. and Zampieri, M. (2017). Detecting hate speech in social media. *arXiv preprint arXiv:1712.06427*.

[Marcus et al., 2002] Marcus, M., Marcinkiewicz, M., and Santorini, B. (2002). Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330.

[McCloskey and Cohen, 1989] McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

[Nielsen, 2011] Nielsen, F. Å. (2011). A new anew: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903*.

[Nikolov and Radivchev, 2019] Nikolov, A. and Radivchev, V. (2019). Nikolov-radivchev at SemEval-2019 task 6: Offensive tweet classification with BERT and ensembles. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 691–695, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

[Pelicon et al., 2019] Pelicon, A., Martinc, M., and Kralj Novak, P. (2019). Embeddia at SemEval-2019 task 6: Detecting hate with neural network and transfer learning approaches. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 604–610, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

[Peters et al., 2018] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

[Poletto et al., 2021] Poletto, F., Basile, V., Sanguinetti, M., Bosco, C., and Patti, V. (2021). Resources and benchmark corpora for hate speech detection: a systematic review. *Language Resources and Evaluation*, 55(2):477–523.

[Ravikiran et al., 2020] Ravikiran, M., Muljibhai, A. E., Miyoshi, T., Ozaki, H., Koreeda, Y., and Masayuki, S. (2020). Hitachi at SemEval-2020 task 12: Offensive language identification with noisy labels using statistical sampling and post-processing. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 961–1967, Barcelona (online). International Committee for Computational Linguistics.

[Reyes et al., 2012] Reyes, A., Rosso, P., and Buscaldi, D. (2012). From humor recognition to irony detection: The figurative language of social media. *Data & Knowledge Engineering*, 74:1–12.

[Rosenthal et al., 2020a] Rosenthal, S., Atanasova, P., Karadzhov, G., Zampieri, M., and Nakov, P. (2020a). A large-scale semi-supervised dataset for offensive language identification. *CoRR*, abs/2004.14454.

[Rosenthal et al., 2020b] Rosenthal, S., Atanasova, P., Karadzhov, G., Zampieri, M., and Nakov, P. (2020b). A large-scale semi-supervised dataset for offensive language identification. *arXiv preprint arXiv:2004.14454*.

[Sachan and Xing, 2016] Sachan, M. and Xing, E. (2016). Easy questions first? a case study on curriculum learning for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 453–463, Berlin, Germany. Association for Computational Linguistics.

[Sanh et al., 2019] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

[Saxena et al., 2019] Saxena, S., Tuzel, O., and DeCoste, D. (2019). Data parameters: A new family of parameters for learning a differentiable curriculum. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

[Schmidt and Wiegand, 2017] Schmidt, A. and Wiegand, M. (2017). A survey on hate speech detection using natural language processing. In *Proceedings of the fifth international workshop on natural language processing for social media*, pages 1–10.

[Shin et al., 2017] Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual learning with deep generative replay.

[Socha, 2020] Socha, K. (2020). KS@LTH at SemEval-2020 task 12: Fine-tuning multi- and monolingual transformer models for offensive language detection. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2045–2053, Barcelona (online). International Committee for Computational Linguistics.

[Sodhani et al., 2019] Sodhani, S., Chandar, S., and Bengio, Y. (2019). Toward training recurrent neural networks for lifelong learning. *Neural Computation*, 32:1–34.

[Sotudeh et al., 2020] Sotudeh, S., Xiang, T., Yao, H.-R., MacAvaney, S., Yang, E., Goharian, N., and Frieder, O. (2020). GUIR at SemEval-2020 task 12: Domain-tuned contextualized models for offensive language detection. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1555–1561, Barcelona (online). International Committee for Computational Linguistics.

[Soviany et al., 2021] Soviany, P., Ionescu, R. T., Rota, P., and Sebe, N. (2021). Curriculum learning: A survey. *arXiv preprint arXiv:2101.10382*.

[Sun et al., 2020] Sun, C., Qiu, X., Xu, Y., and Huang, X. (2020). How to fine-tune bert for text classification?

[Turney and Littman, 2003] Turney, P. D. and Littman, M. L. (2003). Measuring praise and criticism: Inference of semantic orientation from association. *ACM Trans. Inf. Syst.*, 21(4):315–346.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, u., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

[Wan et al., 2020] Wan, Y., Yang, B., Wong, D. F., Zhou, Y., Chao, L. S., Zhang, H., and Chen, B. (2020). Self-paced learning for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1074–1080, Online. Association for Computational Linguistics.

[Wang et al., 2020] Wang, S., Liu, J., Ouyang, X., and Sun, Y. (2020). Galileo at semeval-2020 task 12: Multi-lingual learning for offensive language identification using pre-trained language models. *arXiv preprint arXiv:2010.03542*.

[Wang et al., 2021] Wang, X., Chen, Y., and Zhu, W. (2021). A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP.

[Wang et al., 2019] Wang, Y., Ma, X., Chen, Z., Luo, Y., Yi, J., and Bailey, J. (2019). Symmetric cross entropy for robust learning with noisy labels. pages 322–330.

[Waseem, 2016] Waseem, Z. (2016). Are you a racist or am I seeing things? annotator influence on hate speech detection on Twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 138–142, Austin, Texas. Association for Computational Linguistics.

[Waseem et al., 2017] Waseem, Z., Davidson, T., Warmsley, D., and Weber, I. (2017). Understanding abuse: A typology of abusive language detection subtasks. In *Proceedings of the First Workshop on Abusive Language Online*, pages 78–84, Vancouver, BC, Canada. Association for Computational Linguistics.

[Widmer and Kubat, 1994] Widmer, G. and Kubat, M. (1994). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23.

[Wiedemann et al., 2019] Wiedemann, G., Ruppert, E., and Biemann, C. (2019). UHH-LT at SemEval-2019 task 6: Supervised vs. unsupervised transfer learning for offensive language detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 782–787, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

[Wiegand and Siegel, 2018] Wiegand, M. and Siegel, M. (2018). Overview of the germeval 2018 shared task on the identification of offensive language.

[Wu et al., 2018] Wu, L., Tian, F., Xia, Y., Fan, Y., Qin, T., Lai, J., and Liu, T.-Y. (2018). Learning to teach with dynamic loss functions. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 6467–6478, Red Hook, NY, USA. Curran Associates Inc.

[Wu et al., 2016] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Łukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation.

[Wu et al., 2019] Wu, Z., Zheng, H., Wang, J., Su, W., and Fong, J. (2019). BNU-HKBU UIC NLP team 2 at SemEval-2019 task 6: Detecting offensive language using BERT model. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 551–555, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

[Xu et al., 2020] Xu, B., Zhang, L., Mao, Z., Wang, Q., Xie, H., and Zhang, Y. (2020). Curriculum learning for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6095–6104, Online. Association for Computational Linguistics.

[Xu et al., 2012] Xu, J.-M., Jun, K.-S., Zhu, X., and Bellmore, A. (2012). Learning from bullying traces in social media. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, page 656–666, USA. Association for Computational Linguistics.

[Zampieri et al., 2019a] Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., and Kumar, R. (2019a). Predicting the type and target of offensive posts in social media. pages 1415–1420.

[Zampieri et al., 2019b] Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., and Kumar, R. (2019b). Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval).

[Zampieri et al., 2020] Zampieri, M., Nakov, P., Rosenthal, S., Atanasova, P., Karadzhov, G., Mubarak, H., Derczynski, L., Pitenis, Z., and Çöltekin, Ç. (2020). SemEval-2020 task 12: Multilingual offensive language identification in social media (OffensEval 2020). In *Proceedings of the Fourteenth Workshop on Semantic*

Evaluation, pages 1425–1447, Barcelona (online). International Committee for Computational Linguistics.

[Zhu et al., 2019] Zhu, J., Tian, Z., and Kübler, S. (2019). UM-IU@LING at SemEval-2019 task 6: Identifying offensive tweets using BERT and SVMs. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 788–795, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

[Zhu et al., 2015] Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27.

[Zipf, 2013] Zipf, G. K. (2013). *The psycho-biology of language: An introduction to dynamic philology*. Routledge.