Aesthetic Emotion Classification in Poetry from Small Data

Bachelor thesis by Thanh Tung Linh Nguyen (Student ID: 2809979)
Date of submission: November 16, 2020

Supervisor: Dr. Steffen Eger, Wei Zhao
Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Computer Science
Department

Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 und §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Thanh Tung Linh Nguyen, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß §23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 16. November 2020

Thanh Tung Linh Nguyen

Contents

1. Introduction

1.1. Motivation

Poetry is a form of literature that has been an important part of human language for a long period of time. Poetry differentiates itself from other forms of literature by making use of formal and stylistic constraints such as rhyme or meter.One of poetry's main goals is to deliver emotion experiences to the readers. From the reader's perspective, fully understanding these experiences can be difficult, as there are many ways to perceive the underlying emotions that come with it. Such emotions can be considered *aesthetic emotions*, which are emotions that elicit from a person about an object's aesthetic values when he or she observes or experiences said object (Schindler et al. (2017)). Aesthetic emotions deviate from classical theories of emotions defined by Ekman (1992) in that they are generally more positive as well as more nuanced and fine-grained.

From a Natural Language Processing (NLP) perspective, classifying aesthetic emotions in poetry can be an interesting and challenging task. However, there are very few annotated datasets for aesthetic emotions in poetry available, and these are small in size. There arises a natural question of how to overcome this data sparsity, which we try to address in this thesis with machine learning experiments.

1.2. Goals

Our work has three main goals, which are described as following.

1) Modeling aesthetic emotions in poetry: Emotion classification in poetry can be approached as a multi-class or multi-label classification problem. We can approach this in a poem-, stanza-, or line level, each of which has its own pros and cons. By creating and comparing baseline deep learning models, we can decide which of them is the most suitable for our task.

2) Improving models for aesthetic emotions in poetry: After exploring model architectures, we perform experiments to improve them. The following candidates can be used to make the models more robust in low-resource setting: transfer learning, data augmentation, cross-lingual learning, and ensemble of other approaches.

3) Analysis: Once adequate models for emotion classification in poetry have been trained, an analysis of emotions in poetry should be conducted for author- or period-profiling.

2. Related Work

2.1. Poetry Research in NLP

Poetry research in NLP includes a variety of tasks. Poetry generation, for example, focuses on how to produce meaningful poems that follow aesthetic and formal features. Poetry analysis, on the other hand, uses poetry as a domain for natural language understanding research. Some other works also create poetry datasets with suitable information to facilitate further researches in the field.

In recent times, neural network architectures have been used widely in the field of poetry generation. Lau et al. (2018) introduce Deep-speare, a joint neural network architecture that can generate English sonnets (a style of poem), which have meter and rhyme as formal constraints. Jhamtani et al. (2019) propose an approach to learn rhyming constraints in poetry by using a generative adversarial setup. Liu et al. (2018) use a multi-modal method that generates Chinese poems based on a given theme.

Researches have also been carried out based on poetry's aesthetic features. Reddy and Knight (2011) use an unsupervised algorithm to infer rhyme schemes in English and French poems. Kaur and Saini (2014) include poetry as part of their emotion classification experiment. Alsharif, Alshamaa, and Ghneim (2013) classify emotions for Arabic poems using multiple machine learning approaches. Estes and Hench (2016) buildsa CRF model to classify meter in Middle High German poetry.

Poetry-related corpora are essential for poetry researches, but they are quite limited in number. Sonderegger (2011) applies graph theory to annotate a corpus of English poems with rhyme. The corpus is later expanded with French data. Haider et al. (2020) introduce PO-EMO, a corpus of English and German poems with aesthetic emotion labels. The emotion labels in this dataset include basic emotions categorized by Ekman (1992), as well as more complex and subtle emotions such as *nostalgia, annoyance* and *awe*. We further explore this dataset in Chapter 3, and use it in our experiments.

2.2. Emotion Classification

Emotion classification is an important research field in NLP. As the name suggests, NLP focuses on studying data in the form of texts, which come from multiple sources such as classic literature, newspaper, the Internet. By applying emotion classification, researchers can identify valuable information such as collective reactions to news and events from social media platforms, building automated system for question answering (chatbots), analyzing feedback for user experience gathering, etc.

Given the applications of emotion classification, several datasets from different sources are published and made available. Alm, Roth, and Sproat (2005b) build an annotated corpus based on fairy tales. Aman and Szpakowicz (2007), Liu, Banea, and Mihalcea (2017), Preoţiuc-Pietro et al. (2016) build their corpora on

social media platforms such as Facebook and Twitter. Shmueli and Ku (2019) use extracts from a TV show as part of their work.

A number of currently available emotion corpora annotates their data with 6 labels (*anger, disgust, fear, joy, sadness* and *surpise*) according to Ekman (1992). However, based on their respective evaluations, datasets can contain additional labels such as *no emotion* (Li et al. (2017b), Aman and Szpakowicz (2007)), *mixed emotion* (Aman and Szpakowicz (2007)), *shame* (Ghazi, Inkpen, and Szpakowicz (2015)), *guilt* (Scherer and Wallbott (1994)). Other annotation schemes also exist in contrast to the discrete labels, for example the *Valance-Arousal-Dominance* format (VAD), in which emotions are described using continuous values that span across independent emotion dimensions (Buechel and Hahn (2017)).

While the number of corpora available and their diversity with regard to context are encouraging, the lack of a coherent annotation scheme could be a drawback, as it leads to a limited and restrictive amount of resources available for training a given supervised task. Bostan and Klinger (2018) address this in their work by aggregating a number of datasets using an unified scheme. They also provide an analysis of multiple corpora currently available by comparing them in a systematic way.

As with many other NLP tasks, there are several architectures that can be used to classify emotion in text. Rule-based approaches that rely on feature engineering such as Liew and Turtle (2016), Kunneman, Liebrecht, and Bosch (2014) exist, although they might not be as effective as deep learning based ones. For example, Li et al. (2017a) present a LSTM model together with Skip-gram to classify emotions for short user comments on social media platform. Köper, Kim, and Klinger (2017) also use a CNN-LSTM network, among other features, to predict emotion intensity in tweets.

2.3. Improving Deep Learning Model in Small Data Settings

2.3.1. Transfer Learning

Transfer learning is a method in which knowledge from a domain can be used in a closely related task by relaxing the assumption that the training data and the test data must be independently and identically distributed (Tan et al. (2018)). Pan and Yang (2010) define transfer learning as follow:

Definition 2.3.1. *(Transfer Learning)* Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target predictive function $f_{\mathcal{T}}(.)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_t$ or $\mathcal{T}_S \neq \mathcal{T}_T$.

Base on the availability of $\mathcal{D}_t$, $\mathcal{D}_s$, $\mathcal{T}_t$, and $\mathcal{T}_s$, we can split transfer learning into smaller categories (Pan and Yang (2010)). *Inductive transfer learning* refers to the scenario where the target task is different from the source task, no matter when the source and target domains are the same or not. *Transductive transfer learning* is when the source and target tasks are the same, while the source and target domains are different. *Unsupervised transfer learning* is similar to inductive transfer learning, but there are no labeled data in both source and target domains.

Transfer learning using pre-trained language models, such as word2vec (Mikolov et al. (2013)), ELMo (Peters et al. (2018)), BERT (Devlin et al. (2019)), ELECTRA (Clark et al. (2020)), etc has gained traction in recent years, as it proves to be capable of delivering state-of-the-art performance on a variety of NLP tasks. Typically, language models are trained on a large corpus with unsupervised tasks using deep learning architectures to

learn the language representation, often in the form of word embeddings. In a word embeddings space, words and their meanings are represented by vectors. Words that have similar meanings should be represented by vectors that are close to each other in this space. By fine-tuning these language models with labeled data, we can transfer the knowledge to a specific downstream task. This has proven to be helpful in low-resource setting (Phang, Févry, and Bowman (2019), Pruksachatkun et al. (2020)).

2.3.2. Data Augmentation

Data augmentation can be used to increase the amount of data available in low resources settings by using transformation techniques on currently available data.

Sennrich, Haddow, and Birch (2016) propose *back-translation* as an augmentation technique to improve neural machine translation model. Given a sentence written in a target language, extra training data is obtained by automatically translating the sentence into a source language and pairing the two sentences together. The translation model uses both human-annotated examples and synthetic examples indiscriminately for the training, which shows improvement over using human-annotated examples only. Xie et al. (2019) expand further on the concept to use on text classification tasks, by translating a sentence into a pivot language, and then translate it back to the original language to obtain new examples without losing context. Yu et al. (2018) also use the same concept for training question answering models and point out that translated data could be noisy, which requires careful adjustment when injected into the training process.

Kobayashi (2018) proposes *contextual augmentation* by replacing words with suitable alternatives to create new examples. The substitute candidates are predicted by a bidirectional language model, which shown to be more suitable than swapping with synonyms (Wang and Yang (2015)) or with words from lexical database such as WordNet (Zhang, Zhao, and LeCun (2015)).

Wei and Zou (2019) present EDA (easy data augmentation) consists of 4 different augmentation techniques: *synonym replacement, random insertion, random swap*, and *random deletion*. Only one technique is applied per sentence in the training set, with long sentences are observed to be more noise resistant while still maintaining class label. Even though the operations proposed are simple, they could help to improve results and reduce overfitting in low-resource setting.

2.3.3. Cross-lingual Learning

Cross-lingual learning refers to the usage of data that is available in different languages to help improving performance on a language with fewer resource.

Cross-lingual learning benefits from transfer learning by adapting word embeddings (Section 2.3.1) to build cross-lingual representations of languages. Artetxe and Schwenk (2019) propose a language-agnostic architecture to encode sentences in different languages into fixed-length vectors. Devlin et al. (2019) use a Transformer-based architecture to train m-BERT, a multilingual language model that can represent over 100 languages. m-BERT is pre-trained using the Masked Language Modeling task. Lample and Conneau (2019) and Conneau et al. (2020) expand on this by introducing additional modeling tasks for the pre-training phase. m-BERT, in particular, has proven to be effective at zero-shot learning, which is the process of fine-tuning on one high-resource language before applying it directly on a different, low-resource language (Pires, Schlinger, and Garrette (2019), Wu and Dredze (2019)).

This method has been used in NLP tasks to varying degrees of success. Zoph et al. (2016) transfer learned parameters from a high-resource language pair to a low-resource one to improve neural machine translation (NMT). Kim, Gao, and Ney (2019) use a shared word embeddings space of different languages to improve their NMT model. Xie et al. (2018) apply the same technique to improve named-entity recognition (NER) in low-resource language. Johnson et al. (2019) explore cross-lingual learning to improve NER, using resource from dissimilar language pair (English - Japanese). Karamanolakis, Hsu, and Gravano (2020) use word translations to improve text classification in different language pairs.

2.3.4. Ensemble

For a given machine learning task and a dataset, multiple classifiers can be built. Their predictions can then be combined to provide the final prediction, which in theory should be more correct than the individual predictions. This method is called building an ensemble, which has been proven to be effective. Dietterich (2000) explains three fundamental reasons for this. First, ensemble can reduce the risk of choosing the wrong classifier for a given task. Second, it can provide a better approximation to the optimal classifier than any of individual classifiers. Third, it can find an optimal function f that can represent the optimal classifier correctly.

In our thesis, we are interested in ensembles that can improve multi-label classification, which is the main target of our experiments (Section 3.2). Based on how the multi-label classification task is modeled, there exist different ensembles method that work accordingly (Moyano et al. (2018)).

Multi-label classification can be done by using Binary Relevance (BR), which is a problem transformation technique that builds independent binary classifiers for each label in the dataset, and combine them for a final prediction that does not take the label dependencies into consideration. Ensemble techniques can be used to address this problem. Read et al. (2011) pass down label correlations in a chain of classifiers, while Tsoumakas et al. (2009) apply BR twice, with the second BR layer that takes the output of the first layer.

Another multi-label classification technique is using label powerset (LP), in which each possible label combination in the training data is treated as an unique label. Classifiers are then built to predict these label combinations. The disadvantage of LP is that some label combinations may appear very infrequently, which makes it hard to predict. Tsoumakas and Vlahavas (2007) randomly separate the label set into smaller size set, build classifiers on those sets and combine their predictions by voting to overcome this. Rokach, Schclar, and Itach (2014) improve on this by thresholding the probabilities output by the classifiers instead of voting. The threshold is calculated based dataset, instead of being set manually.

Nguyen et al. (2020) describe a framework to combine output of classifiers. The framework includes two classifiers combining strategies: predict-then-combine (PTC) and combine-then-predict (CTP). For first strategy, the classifiers provide their binary predictions for a sample, which are then combined on a label basis to make the final prediction. The second strategy uses the probabilities output by the participating classifiers and combine them into a single representation, which is used to make the final prediction.

3. Dataset

For our emotion classification experiments, we use the PO-EMO dataset (Haider et al. (2020)) as the main resource. In this chapter, we will explain the dataset in detail with regard to the annotation, poem distribution and label distribution. In some experiments, additional datasets are also used, which we will discuss in their relevant sections.

Text	Annotator 1	Annotator 2
Music , when soft voices die ,	Beauty / Joy — Sadness	Sadness
Vibrates in the memory —	Beauty / Joy — Sadness	Sadness
Odours , when sweet violets sicken ,	Beauty / Joy — Sadness	Sadness
Live within the sense they quicken .	Beauty / Joy — Sadness	Sadness
Rose leaves , when the rose is dead ,	Sadness — Beauty / Joy	Sadness
Are heaped for the beloved ”s bed ;	Sadness — Beauty / Joy	Sadness
And so thy thoughts , when thou art gone ,	Sadness — Beauty / Joy	Sadness
Love itself shall slumber on .	Sadness — Beauty / Joy	Sadness

Table 3.1.: Example: A poem annotated by two annotators in PO-EMO. If a line is annotated by with more than one label by an annotator, they are separated by '—'. The poem consists of two *stanza*, which are separated by a blank line.

3.1. Overview

PO-EMO consists of 158 German poems and 64 English poems, which is small in size compared to other emotion classification datasets. All poems are annotated with aesthetic emotion labels by experts, following specific annotation guidelines. The poems in PO-EMO can be categorized into three groups: Gold standard poems, English poems and German poems. There are 48 gold standard poems, which are annotated by three experts. Each expert annotates the lines with emotion labels, and the final annotation for each line is decided by majority voting. English and German poems are annotated by only two experts, using the gold standard poems as part of the guidelines. The annotators have agreement levels ranging from 0.5 to 0.8 Cohen's kappa, depending on the label. For our baseline emotion classification tasks, we do not use the gold standard poems and only use the English poems.

Each line in the poems is annotated with the following emotion labels: *Suspense, Awe/Sublime, Sadness, Annoyance, Uneasiness, Beauty/Joy, Vitality, Humor* and *Nostalgia*. Only English poems have the label *Nostalgia*. As noted by Haider et al. (2020), the emotion labels here represent the reader perspective, i.e how the readers feel after reading the lines, rather than the hypothetical emotions that the poets want to deliver.

Each annotator can assign a line with up to two labels, which adds up to a maximum of 4 labels per line for poems outside of the Gold standard. Example of an English poem from the dataset can be seen in Table 3.1. Annotators are encouraged to maintain a degree of consistency by annotating the lines in the context of the stanzas or poems they belong to. They should also follow the gold standard when in doubt.

As seen in Table 3.2a, the numbers of lines per label are not equal. Labels such as *Beauty/Joy, Sadness* and *Vitality* have a lot samples in the dataset; while *Nostalgia*, for example, rarely appears. Despite the maximum of 4 unique labels per line, the majority of lines only have one or two labels (Table 3.2b), which is reflective of the relatively high agreement score.

Label	No. of Lines
Suspense	126
Awe/Sublime	143
Sadness	394
Annoyance	50
Uneasiness	269
Beauty/Joy	540
Vitality	340
Humor	76
Nostalgia	58

(a) Number of lines per each label.

No. of unique Label	No. of Lines
1	532
2	654
3	52
4	0

(b) Number of unique labels per line.

Table 3.2.: Statistics for line-level annotation scheme. We aggregate the annotation from both annotators. In case they use the same label, we count the line as one instance only.

3.2. Preprocessing

According to the annotation guidelines for PO-EMO, even though each line is annotated individually, it is important to look at them with regard to their appropriate context, i.e stanza or poem as well. Therefore, we model our classification task as stanza-level emotion classification, and preprocess the data accordingly. The reason why we choose stanza and not poem level is because we feel that in a stanza, emotions are more consistent and thus easier to classify than in a poem. We extract 174 stanzas from the English poems and 512 stanzas from the German poems.

For each stanza, we aggregate the emotion labels from the lines within it to get the stanza-level emotion labels. By doing this, a stanza can have more than one emotion label. We take the annotations from both annotators. If they agree on one emotion label, that label will only count as one instance for the stanza-level label. The first stanza in Table 3.1 has the aggregated label of *[Beauty / Joy, Sadness]*, and the second one has *[Sadness, Beauty / Joy]*.

From the original 9 labels (*Suspense, Awe/Sublime, Sadness, Annoyance, Uneasiness, Beauty/Joy, Vitality, Humor, Nostalgia*), we remove *Nostalgia* from the annotation. This label is used quite infrequently and always annotated together with other labels, making it difficult to classify properly. There are no stanza that had nostalgia as the only label, so after removing all Nostalgia label, all stanzas still have at least one label and thus doesn't lower the total amount of data. We report the number of stanzas which have a certain label for English and German stanzas in Table 3.3.

We randomly separate the English stanzas into two splits, which we call **split_0** and **split_1**. Table 3.4 shows the label distribution in our two splits. We model our task as a multi-label classification task by using these two splits. For each experiment, we train two models: One trained on split_0 and tested on split_1; one trained on split_1 and tested on split_0. We report the average F1-score for each label, as well as the average F1-macro score in our experiments.

Label	English	German
Suspense	24	70
Awe/Sublime	30	64
Sadness	67	149
Annoyance	11	25
Uneasiness	34	158
Beauty/Joy	78	233
Vitality	50	103
Humor	17	49

Table 3.3.: Label distribution of English and German stanzas.

Label	split_0	split_1
Suspense	13	11
Awe/Sublime	13	17
Sadness	34	33
Annoyance	5	6
Uneasiness	14	20
Beauty/Joy	39	39
Vitality	22	28
Humor	10	7

Table 3.4.: Label distribution of stanzas in split_0 and split_1.

4. Baseline Models

In this chapter, we explain the training of baseline models for multi-label emotion classification in poetry in detail. We consider three approaches to this task:

- Training a model using information from a lexicon by Mohammad (2018).
- Training a neural network model using the Transformers-based BERT architecture Devlin et al. (2019).
- Training a neural network model using a BiLSTM-CNN-CRF architecture by Reimers and Gurevych (2017).

4.1. Lexicon-based Model

Displaying emotions using a set of discrete labels is not the only way to represent the meaning of words. The Valence-Arousal-Dominance (VAD) model, for example, offers a way to model emotions by measuring their intensity across independent categories. Buechel and Hahn (2017) described the 3 categories of this model as follows:

- Valence: how positive one feels towards a situation or an object.
- Arousal: how excited one feels towards a situation or an object.
- Dominance: how in control one feels in a situation.

The VAD model measures words in a scale between 0.0 and 1.0 across these 3 dimensions, with 0.0 being the lowest (one is feeling completely negative, calm or controlled) and 1.0 the highest (one is feeling completely positive, excited or in control). An illustration of the model can be found in Figure 4.1.

By using the VAD values, we can look at the emotions of words in a continuous way, rather than being restricted by a predetermined set of labels. This could provide a better understanding of the nuances of emotions. To incorporate the VAD model into our task, the NRC-VAD lexicon by Mohammad (2018) is used. This lexicon contains more than 20,000 commonly used English words, annotated by crowd-sourcing.

For each stanza in the PO-EMO dataset, we calculate the following metrics: average values of all words in the stanza, the maximum value of all words and the minimum value of all words (this minimum value should be greater than 0). We extract the 3 metrics above for the Valence dimension, the Arousal dimension and the Dominance dimension. After that, we train multi-label classifiers with the two splits as described in Section 3.2. We build the classifiers using the scikit-learn library[1]. The classifiers are implemented with the Binary Relevance strategy, in which the multi-label task is reduced to smaller binary classification tasks for each label. We use Linear Support Vector Classification[2] for the binary classifiers. For a text, each binary classifier

[1] http://scikit.ml/index.html
[2] https://scikit-learn.org/stable/modules/svm.html

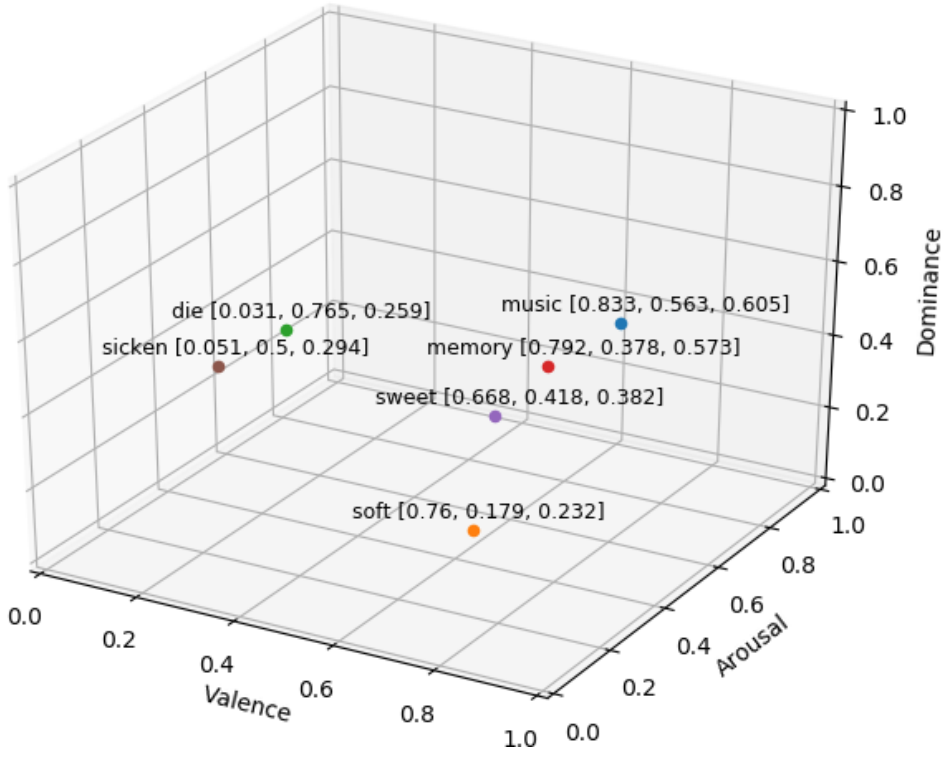


Figure 4.1.: Example: VAD Model. Words are randomly taken from Table 3.1

provides a binary prediction for its respective label. The final multi-label prediction is the union of these predictions.

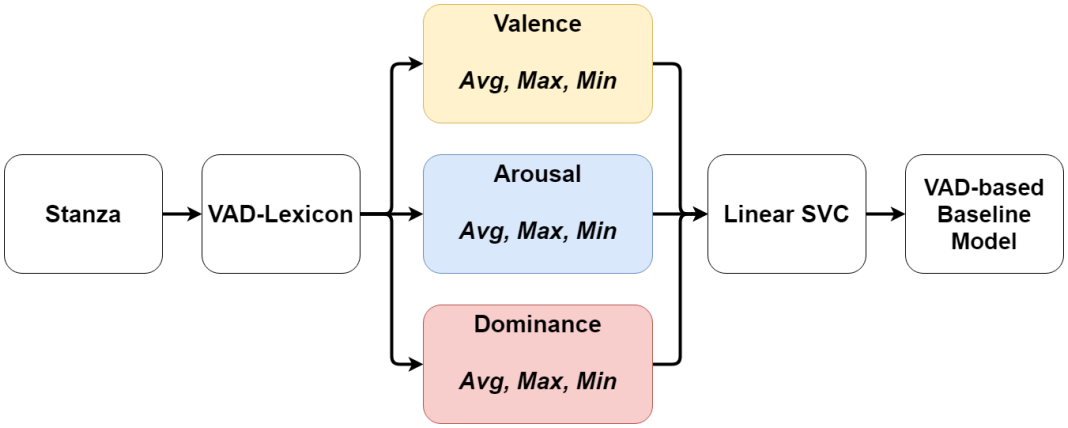


Figure 4.2.: Building the VAD-based model

The results for this approach can be seen in Table 4.1. Overall, the scores are not very impressive, and some of the labels could not be predicted properly, judging from their low F1-scores.

Label	0/1	1/0	Average
Suspense	0.00	0.11	0.06
Awe/Sublime	0.36	0.36	0.36
Sadness	0.57	0.59	0.58
Annoyance	0.14	0.19	0.17
Uneasiness	0.49	0.44	0.47
Beauty/Joy	0.68	0.70	0.69
Vitality	0.46	0.41	0.43
Humor	0.08	0.17	0.13
F1-macro Score	0.35	0.37	0.36

Table 4.1.: F1-score for the baseline Lexicon-based models.
0/1: model trained on split_0 and tested on split_1. **0/1**: model trained on split_1 and tested on split_0.

4.2. BERT-based Model

BERT, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers, is a Transformers-based architecture by Devlin et al. (2019). BERT is built on a multi-layer bidirectional Transformer encoder with attention mechanism (Vaswani et al. (2017)). Ever since its publication, BERT has achieved state-of-the-art performance on a multiple NLP tasks, including text classification. BERT follows a two-step framework: *pre-training* and *fine-tuning*.

BERT is pre-trained on a large corpus by using two unsupervised tasks. The first task is Masked Language Modeling (MLM), which masks part of the input tokens at random, and then predict those masked tokens.

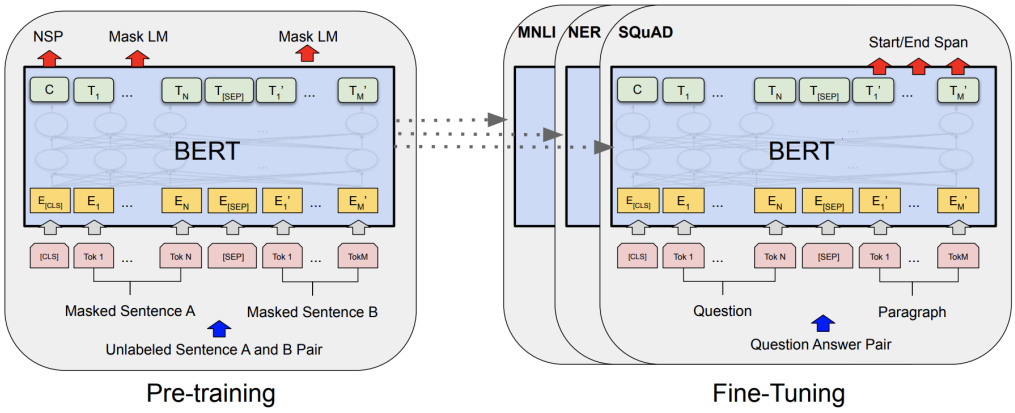


Figure 4.3.: BERT by Devlin et al. (2019). The framework consists of two steps: *Pre-training* on a large corpus with the MLM and NSP tasks, and *fine-tuning* to make the model more suitable to a variety of downstream tasks

The other pre-training task is Next Sentence Prediction (NSP), which contributes to the model's understanding of sentence relationships in a corpus. After the pre-training steps, BERT can be initialized by loading all the pre-trained parameters. A list of pre-trained BERT model and its variants can be found in `https://huggingface.co/transformers/pretrained_models.html`. Pre-trained BERT models can then be loaded and further *fine-tuned* for a specific downstream task by providing appropriate input data and a classification layer.

In this thesis, we use BERT-large, a variant of BERT. BERT-large is fine-tuned with more parameters than the base version of BERT, although it can be unstable when used on small datasets as noted by Devlin et al. (2019). We fine-tune[3] BERT-large with the stanzas from PO-EMO. The input stanzas are represented by concatenating the lines together into one line, separated by special tokens (</br>) between each lines. The labels for each stanza are represented by a binary vector. We add a classification layer on top of BERT-large and train it for multi-label classification. The loss function for the layer is Binary cross-entropy with logits. A sigmoid activation function is used on the logits to get the probabilities for each label. We threshold the probabilities to receive the binary prediction for all labels and represent them as a final binary vector. For the baseline approach, we use a simple threshold of 0.5. Table 4.2 shows the result for fine-tuning BERT approach.

4.3. BiLSTM-CNN-CRF Model

The two baseline approaches above have looked at stanzas in an independent manner, i.e we assumed that emotions in one stanza have no effect on the stanzas that follow or precede it. However, this might not be appropriate in poetry emotion classification, since emotions in one stanza can be influenced by its surrounding context. Therefore, building a model that takes *label dependencies* between constituents into consideration could be beneficial to our task.

[3]BERT fine-tuning code is provided by SimpleTransformers: `https://github.com/ThilinaRajapakse/simpletransformers`

Label	0/1	1/0	Average
Suspense	0.43	0.00	0.21
Awe/Sublime	0.00	0.30	0.15
Sadness	0.59	0.57	0.58
Annoyance	0.00	0.00	0.00
Uneasiness	0.32	0.46	0.39
Beauty/Joy	0.74	0.75	0.75
Vitality	0.55	0.53	0.54
Humor	0.44	0.18	0.31
F1-macro Score	0.38	0.35	0.37

Table 4.2.: F1-score for the baseline BERT models. **0/1**: model trained on split_0 and tested on split_1. **0/1**: model trained on split_1 and tested on split_0.

Recurrent Neural Network (RNN) and its variants have shown their effectiveness in capturing dependencies within sequences (Wu et al. (2016), Hu et al. (2016)). Ma and Hovy (2016) proposed RNN-based neural architecture that can make use of both word-level and character-level representation to tackle sequence labeling tasks. Their architecture is a combination of bidirectional LSTM (Long-Short Term Memory), CNN (convolutional neural network) and CRF (conditional random fields). It uses character-level representation generated by the CNN, combined with word embeddings as word-level representation as input for the BiLSTM. The output vectors of the BiLSM are then decoded by the CRF layer to figure out the best sequence.

Using the implementation of this architecture by Reimers and Gurevych (2017), we tag the lines in stanzas with appropriate labels. The tagged labels can then be aggregated as described in Chapter 3 to get the stanza emotion, which is crucial since we want to compare the performance between approaches. We generate the word embeddings representation for each line using LASER (Artetxe and Schwenk (2019)) and treat each line as a constituent to the stanza by concatenating the words with '_' symbol. We use a LSTM size of 1024 to match the dimension of LASER embeddings, and a character embeddings size of 500. The models are optimized using F1-macro score, and the results can be seen in Table 4.3 .

Label	0/1	1/0	Average
Suspense	0.00	0.20	0.10
Awe/Sublime	0.00	0.00	0.00
Sadness	0.39	0.42	0.41
Annoyance	0.29	0.00	0.14
Uneasiness	0.44	0.33	0.39
Beauty/Joy	0.48	0.41	0.45
Vitality	0.43	0.83	0.63
Humor	0.35	0.75	0.55
F1-macro Score	0.30	0.23	0.26

Table 4.3.: F1-score for the baseline BiLSTM-CNN-CRF models. **0/1**: model trained on split_0 and tested on split_1. **0/1**: model trained on split_1 and tested on split_0.

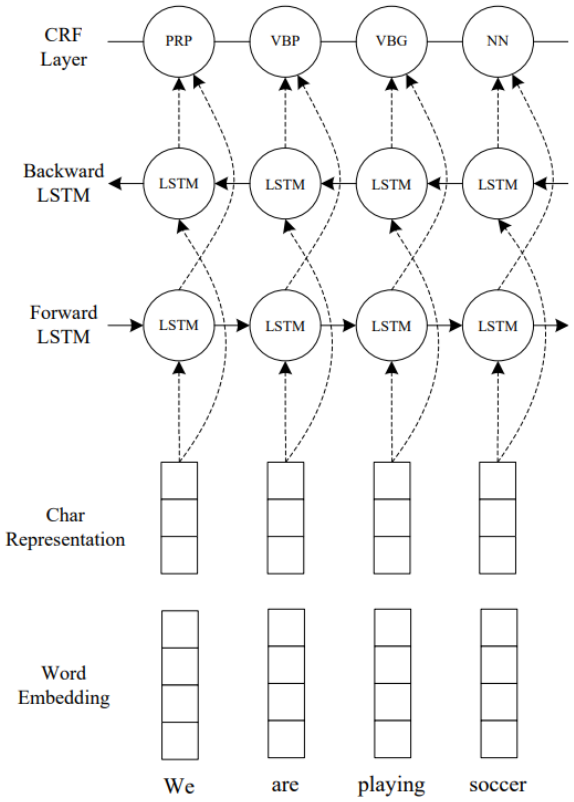


Figure 4.4.: BiLSTM-CNN-CRF architecture by Ma and Hovy (2016). The character-level representation is encoded by an convolutional neural network and concatenated to the word embeddings representation of a sentence. The BiLSTM network then takes this concatenated representation as input. Conditional Random Fields (CRF Layer) process the output of the BiLSTM network to make the final prediction.

4.4. Conclusion

To establish a baseline of comparison for our task, three approaches were considered: Lexicon-based, BERT-based and BiLSTM-CNN-CRF architecture. They all have their pros and cons, and each looks at the classification task in different ways. Overall, the baseline results (as seen in Figure 4.4 and 4.5) leave much room for improvement, as expected from the low amount of data available. While the results from the first two approaches are close to each other, the last one falls behind noticeably (-10 point in F1-macro score). On a per-label basis, the BERT-based and BiLSTM approaches have difficulties in classifying labels that are underpopulated in the dataset. The score for each labels correlate strongly with the amount of data available. As the BERT-based approach has the best performance, we use it as the baseline of comparison for our experiments.

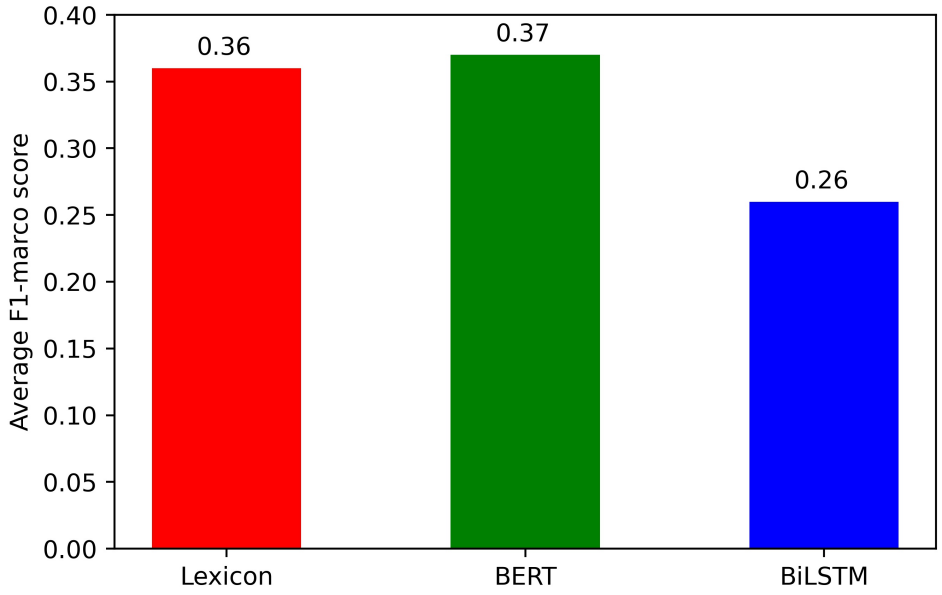


Figure 4.5.: Baseline models: Average F1-macro scores.
Lexicon: Lexicon-based models. **BERT**: BERT-based models. **BiLSTM**: BiLSTM-CNN-CRF models.

Figure 4.6.: Baseline models: F1 scores per label.
Lexicon: Lexicon-based models. **BERT**: BERT-based models. **BiLSTM**: BiLSTM-CNN-CRF models.

5. Experiments

5.1. Transfer Learning

Transfer learning is the process of transferring knowledge between different, but related tasks or domains to improve performance of a model. This could potentially alleviate the problem of small data in our task, by training on tasks with more data available, and then apply transfer learning to our current task.

The BERT language model by Devlin et al. (2019) makes use of this technique by pre-training the base model with unsupervised tasks on a large amount of unlabeled data. The pre-trained model can then be fine-tuned using labeled data for specific downstream NLP tasks. Phang, Févry, and Bowman (2019) further expand on this idea by suggesting an approach called *Supplementary Training on Intermediate Labeled-data Tasks (STILTs)*, which adds a second stage of pre-training with an intermediate supervised task before the target downstream task. STILTs can improve the overall performance and reliability of the downstream model, particularly when the target task only has a small amount of training data available.

We adapt both fine-tuning BERT and STILTs in our experiments using the two-step setting shown in Figure 5.1. Using BERT-large as the base model, we fine-tune it with either unsupervised (Section 5.1.1) or supervised (Section 5.1.2, Section 5.1.3) tasks, before fine-tuning it with labeled data from PO-EMO for our classification task. After that, we compare the results to see if there is any improvement over the baseline results. We perform the following three experiments: *poetry fine-tuning, emotion fine-tuning* and *meter fine-tuning*.

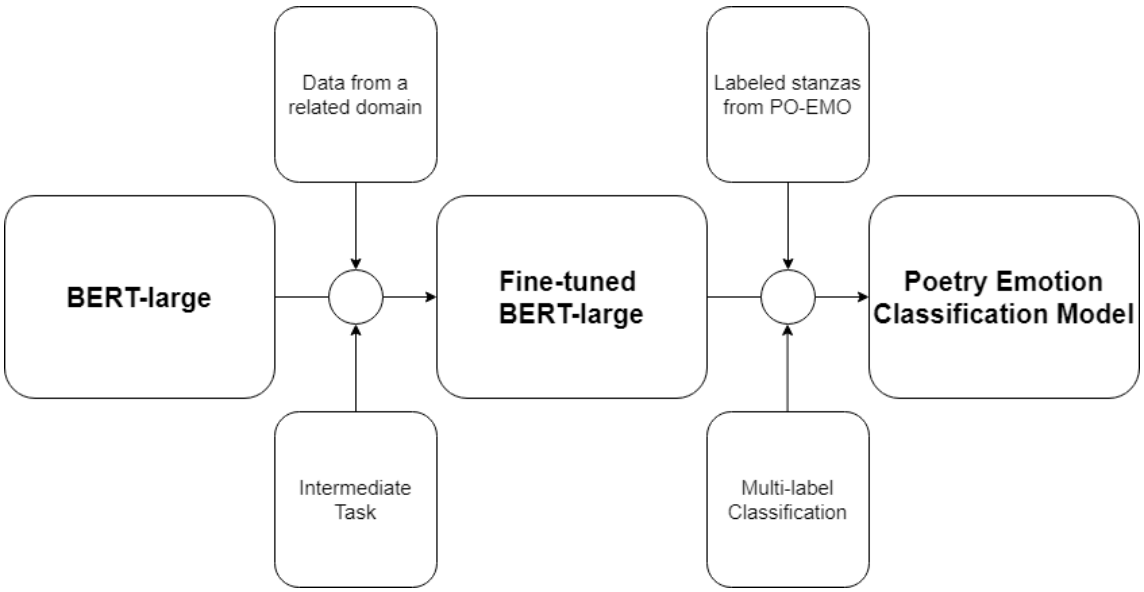


Figure 5.1.: Transfer learning pipeline with pre-trained BERT-large.

5.1.1. Poetry Fine-tuning

Our first experiment is to fine-tune pre-trained BERT-large with more poetry data to help it understand the more detailed structure of the language used in our task. The pre-trained BERT model is trained on the BooksCorpus (800M words) and English Wikipedia (2,500M), which contributes to its broad understanding of the English language. However, fine-tuning BERT can be less effective when the downstream tasks use specialized language that it is not familiar with. This problem can be addressed by further pre-training BERT on domain-specific data before applying it to the downstream task, which can be seen as a new BERT-based model that has a better understanding of the language usage in that domain. Examples for these domain-specific BERT variants include SciBERT by Beltagy, Cohan, and Lo (2019) fine-tuned on 3.17B words for scientific publications; BioBERT by Lee et al. (2019) fine-tuned on 18B words for biomedical text and BERTweet by Nguyen, Vu, and Nguyen (2020) fine-tuned on 850M English tweets, all of which outperform baseline BERT-model in their respective domain-related NLP tasks.

Following the above examples, we fine-tuned BERT-large with a small amount of unlabeled poetry data to improve our model's performance in our classification task. We randomly sample 30,000 English stanzas from poems in the Gutenberg English Poetry Corpus[1] to fine-tune the model. These stanzas are not labeled with emotions. We concatenate the lines in each stanza into one single line, with each line separated by a special token. As the stanzas are randomly chosen and thus not suitable for the Next Sentence Prediction task, we only use the unsupervised Masked Language Modeling task in the fine-tune process. We then use the resulting model to classify emotion with the same classify layer described in Section 4.2. The result can be seen in Table 5.1.

Label	0/1	1/0	Average
Suspense	0.29 (- 0.14)	0.33 (+ 0.33)	0.31 (+ 0.1)
Awe/Sublime	0.52 (+ 0.52)	0.52 (+ 0.22)	0.52 (+ 0.37)
Sadness	0.63 (+ 0.04)	0.63 (+ 0.07)	0.63 (+ 0.05)
Annoyance	0.25 (+ 0.25)	0.33 (+ 0.33)	0.29 (+ 0.29)
Uneasiness	0.56 (+ 0.23)	0.50 (+ 0.05)	0.53 (+ 0.14)
Beauty/Joy	0.74 (+ 0.00)	0.67 (- 0.08)	0.71 (- 0.08)
Vitality	0.63 (+ 0.07)	0.40 (- 0.13)	0.51 (+ 0.25)
Humor	0.57 (+ 0.13)	0.43 (+ 0.25)	0.50 (+ 0.19)
F1-macro Score	0.52 (+ 0.14)	0.48 (+ 0.13)	0.50 (+ 0.13)

Table 5.1.: F1-score for the emotion classification BERT models based on poetry fine-tuned BERT-large. **0/1**: model trained on split_0 and tested on split_1. **0/1**: model trained on split_1 and tested on split_0. Score differences compared to baseline BERT model are given in ( )

After fine-tuning with poetry, our emotion classification models show better understanding of the *Suspense, Awe/Sublime* and *Annoyance* labels, which the baseline models fail to classify adequately. The only label which our poetry fine-tuned models show less effectiveness in classifying is Beauty/Joy, but not by much. The experiment shows an increase in overall performance compared to the baseline BERT models with a 0.50 average F1-macro score (up from 0.37 average), which is consistent with results from other domain-specific BERT variants. Our experiment achieves surprisingly good results, despite being fine-tuned with much less data compared to these variants. One possible explanation for the result is that, since our downstream classification task is low-resource in nature, introducing more poetry data into BERT vocabulary has a more

[1] `https://github.com/tnhaider/english-gutenberg-poetry`

visible impact on the final result. We could further investigate this in future works by fine-tuning BERT with more than 30,000 samples and see if the trend of increasing performance continues.

5.1.2. Emotion Fine-tuning

We can also increase the model's understanding of emotion classification by using intermediate task training. For this experiment, we choose emotion classification in fairy tales as the intermediate task for a number of reasons. Firstly, like poetry, fairy tales contain a lot of immersive and subtle emotions, from common ones like happy or sadness to more nuanced ones like disgusted or fearful. Secondly, as the language used in poetry can have different formal restrictions or meanings compared to everyday conversations, it is important to choose a domain that is close to our main task. Fairy tales can be a suitable domain for this, as they are both parts of classical literature. Thirdly, the stories depicted in fairy tales cover a wide range of contexts, which is also one of the characteristics of poetry.

Our fairy tales data comes from the *tales-emotion* corpus by Alm, Roth, and Sproat (2005a) and Alm (2008) with over 14,000 lines from fairy tales written by 3 well-known authors: B. Potter, H.C. Andersen and the Grimm brothers. In this corpus, each line in the stories is annotated with the following emotion labels, some of them are similar to the PO-EMO annotation: *Angry, Disgusted, Fearful, Happy, Neutral, Sad, Positively Surprised* and *Negatively Surprised* (See Table A.1 for more details). There is an inherent imbalance in the dataset, as there are a lot of samples with *Neutral* or *Happy* labels, and not a lot of samples with *Disgusted* or *Fearful*. Like PO-EMO, tales-emotion is annotated by two annotators. They only agree on 1,280 of these sentences, with Cohen's kappa agreement lies between 0.24 and 0.51 for each label. This corpus is reviewed in the context of emotion classification by Bostan and Klinger (2018), in which they use the corpus in two scenarios: within-corpus emotion classification with cross-validation and cross-corpus emotion classification (train on the entire data of a corpus, and use it to classify data of other corpora). Using maximum entropy classifiers with Bag of Words as features, they achieve a macro F1-score of 0.54 in the first scenario. The corpus is also considered "informative" by the authors in the second scenario, where training on tales-emotion leads to good classification results on other corpora. This supports our idea of using this corpus for intermediate task training.

We model the intermediate task as a multi-label classification one, using the tales-emotion corpus with labels assigned by both annotators. Unlike our downstream classification setup for poetry, we only take individual lines as input, as paragraphs (the equivalence of stanzas in poems) can be too long and exceed the technical limitations of BERT[2]. We randomly take 10,000 lines for training and leave the rest for testing. The result for the intermediate task can be seen in Table 5.2a. We then use the resulting model for our downstream poetry classification task with a multi-label classification layer, the result of which can be seen in Table 5.2b.

Our emotion classification model for fairy tales performs well on labels that are common in tales-emotion corpus, such as *Happy, Neutral* and *Sad*. However, this doesn't seem to help when this knowledge is transferred to the downstream task, as the F1-scores for the closely related *Beauty/Joy* and *Sadness* labels reduce by a slight amount. The models also completely fail to classify *Annoyance*, which is one of the main problems with the baseline models. On the other hand, we have some improvements for labels such as *Awe/Sublime* and *Uneasiness*. *Humor*, in particular, goes from 0.31 in average F1-score to 0.68 average F1-score. Compared with the more successful fine-tuning approach in Section 5.1.1, this experiment has two differences: less training data for the intermediate task and less closely related dataset. Nevertheless, the macro F1-score

[2]Maximum sequence length of 512 tokens

Label	F1-Score
Angry	0.50
Disgusted	0.32
Fearful	0.44
Happy	0.58
Neutral	0.94
Sad	0.58
Positively Surprised	0.28
Negatively Surprised	0.30
F1-macro Score	0.49

(a) F1-score for multi-label emotion classification with fairy tales lines. Trained on 10,000 lines, tested on 4,208 lines.

Label	0/1	1/0	Average
Suspense	0.20 (- 0.23)	0.21 (+ 0.21)	0.21 (- 0.01)
Awe/Sublime	0.24 (+ 0.24)	0.35 (+ 0.05)	0.29 (+ 0.14)
Sadness	0.57 (- 0.02)	0.51 (- 0.05)	0.54 (- 0.03)
Annoyance	0.00 (+ 0.00)	0.00 (+0.00)	0.00 (+ 0.00)
Uneasiness	0.51 (+ 0.20)	0.44 (- 0.01)	0.48 (+ 0.09)
Beauty/Joy	0.64 (- 0.10)	0.61 (- 0.14)	0.63 (- 0.12)
Vitality	0.65 (+ 0.10)	0.50 (- 0.03)	0.58 (+ 0.04)
Humor	0.77 (+ 0.33)	0.59 (+ 0.41)	0.68 (+ 0.37)
F1-macro Score	0.45 (+ 0.06)	0.40 (+ 0.05)	0.43 (+ 0.06)

(b) F1-score for the emotion classification BERT models with fairy tales emotion classification as intermediate task. **0/1**: model trained on split_0 and tested on split_1. **0/1**: model trained on split_1 and tested on split_0. Score differences compared to baseline BERT model are given in ( ).

Table 5.2.: Result: Intermediate task training with emotion fine-tuning.

increases from 0.37 for the baseline models to 0.43, which shows that intermediate task training can help us improve the baseline models.

5.1.3. Meter Fine-tuning

Our third transfer learning experiment uses meter, a domain which is closely related to emotion in poetry, as the domain for transfer learning. Meter is one of the formal constraints of poetry, which also contributes to the overall aesthetic of the art form. It describes the rhythmic structure of a poetry line by denoting the stress of syllables within it. Meter usually follows certain patterns that repeat multiple times in a stanza or the poems. Reading a line by following the indicated meter patterns can usually result in different expressions and intonations compared to how it would be expressed in normal conservation. A study by Obermeier et al. (2013) shows evidence that meter (and rhyme) could contribute to the aesthetic and emotional evaluation of poems.

Like the second transfer learning experiment, we also apply intermediate task training here. Our intermediate task in this case is meter classification. Our dataset[3] consists of 1,412 poetry lines annotated with detailed meter patterns. We simplify the patterns with regular expressions, for example a pattern of '-+|-+|-+|-+|-+' would translate to 'iambic.penta', which means the iambic foot '-+' repeats 5 times (penta). An example of a stanza with simplified meter annotation can be found in 5.3. There is a total of 13 simplified meter patterns in the dataset. However, since there are some patterns with only 1 or 2 samples, we combine them into a new category 'other'. The final set of label consists of *iambic.di, iambic.tri, iambic.tetra, chol.iamb, alexandr.iambic.hexa, troch.tetra, iambic.penta* and *other*. Since each line can only have one meter scheme, the classification task is single-label in nature. Afterwards, we use the resulting model for our poetry emotion classification task. The result for our intermediate task and downstream task can be seen in Table 5.4a and Table 5.4b, respectively.

Together with decent score increases in less common labels such as *Annoyance* or *Humor*, our average macro F1-score increases from 0.37 to 0.48, which is promising. With the same intermediate training setup, meter fine-tuning outperforms the emotion fine-tuning experiment, despite having less data used in the task (1,412

[3] `https://github.com/manexagirrezabal/for_better_for_verse`

Text	Meter pattern	Simplified pattern
the brain–is wider than the sky	-+|-+|-+|-+|	iambic.tetra
for– put them side by side–	-+|-+|-+|	iambic.tri
the one the other will contain	-+|-+|-+|-+|	iambic.tetra
with ease– and you–beside–	-+|-+|-+|	iambic.tri

Table 5.3.: Example: A stanza annotated with meter pattern. + signifies an unstressed syllable, - signifies a stressed syllable and | signifies a foot boundary to group the rhythmic units.

Label	F1-Score
iambic.di	0.46
iambic.tri	0.56
iambic.tetra	0.55
chol.iamb	0.10
alexandr.iambic.hexa	0.38
troch.tetra	0.00
iambic.penta	0.80
other	0.29
F1-macro Score	0.39

(a) F1-score for multi-class meter classification in poetry. Trained on 1,000 lines, tested on 412 lines.

Label	0/1	1/0	Average
Suspense	0.10 (- 0.33)	0.36 (+ 0.36)	0.23 (+ 0.02)
Awe/Sublime	0.45 (+ 0.45)	0.39 (+0.09)	0.42 (+ 0.27)
Sadness	0.61 (+ 0.02)	0.63 (+ 0.06)	0.62 (+ 0.04)
Annoyance	0.20 (+ 0.20)	0.33 (+ 0.33)	0.27 (+ 0.27)
Uneasiness	0.53 (+ 0.21)	0.46 (+ 0.01)	0.50 (+ 0.11)
Beauty/Joy	0.70 (- 0.05)	0.66 (- 0.09)	0.68 (- 0.07)
Vitality	0.60 (+ 0.04)	0.45 (- 0.08)	0.52 (- 0.02)
Humor	0.56 (+ 0.11)	0.59 (+ 0.41)	0.57 (+ 0.26)
F1-macro Score	0.47 (+ 0.08)	0.48 (+ 0.14)	0.48 (+ 0.11)

(b) F1-score for the emotion classification BERT models with meter classification as intermediate task. **0/1**: model trained on split_0 and tested on split_1. **0/1**: model trained on split_1 and tested on split_0. Score differences compared to baseline BERT model are given in ( ).

Table 5.4.: Result: Intermediate task training with meter fine-tuning.

samples versus 14,208 samples). This reinforces our hypothesis in Section 5.1.1 that introducing extra poetry data before the final fine-tuning task could help BERT understand the language used in poetry better. This also opens up the possibility of including other aesthetic features of poetry, such as rhyme, for the intermediate task. However, the large difference in training data used (30,000 samples versus 1,412 samples) suggests that further investigation might be needed, as to which method of introducing between fine-tuning or intermediate task training is more effective.

5.1.4. Additional Transfer Learning Experiments

Following the success we have with BERT-based transfer learning methods, we experiment with combining fine-tuning with unlabeled data and intermediate task training. Instead of using pre-trained BERT-large model as the starting (or base) model for intermediate task training, we use our BERT-large model that has been fine-tuned with poetry from Section 5.1.1. We perform two experiments with this method. The first experiment uses fairy tales emotion classification (Section 5.1.2) as the intermediate task, and the second uses meter classification (Section 5.1.3) as the intermediate task. After intermediate task training, we train the resulting models for poetry emotion classification, and report the results in Table 5.5 and Table 5.6.

Contrary to our expectations, the results show no improvements when compared to using pre-trained BERT-large as starting model for intermediate task training (Table 5.7a and Table 5.7b). While the two methods work well independently, combining them doesn't seem to improve current results. We have no answer to

Label	0/1	1/0	Average
Suspense	0.08 (- 0.35)	0.22 (+ 0.22)	0.15 (- 0.06)
Awe/Sublime	0.31 (+ 0.31)	0.14 (- 0.16)	0.22 (+ 0.07)
Sadness	0.54 (- 0.05)	0.55 (- 0.02)	0.54 (- 0.03)
Annoyance	0.00 (+ 0.00)	0.00 (+ 0.00)	0.00 (+ 0.00)
Uneasiness	0.53 (+ 0.21)	0.35 (- 0.11)	0.44 (+ 0.05)
Beauty/Joy	0.60 (- 0.14)	0.62 (- 0.12)	0.61 (- 0.13)
Vitality	0.51 (- 0.04)	0.59 (+ 0.06)	0.55 (+ 0.01)
Humor	0.80 (+ 0.36)	0.57 (+ 39)	0.69 (+ 0.30)
F1-macro Score	0.42 (+ 0.04)	0.38 (+ 0.03)	0.40 (+ 0.03)

Table 5.5.: F1-score for the emotion classification BERT models with poetry fine-tuned BERT as starting model and fairy tales emotion classification as intermediate task.
0/1: model trained on split_0 and tested on split_1. **0/1**: model trained on split_1 and tested on split_0. Score differences compared to baseline BERT model are given in ( ).

Label	0/1	1/0	Average
Suspense	0.31 (- 0.12)	0.47 (+ 0.47)	0.39 (+ 0.17)
Awe/Sublime	0.29 (+ 0.29)	0.40 (+ 0.10)	0.34 (+ 0.19)
Sadness	0.55 (- 0.03)	0.60 (+ 0.03)	0.58 (+ 0.00)
Annoyance	0.22 (+ 0.22)	0.29 (+ 0.29)	0.25 (+ 0.25)
Uneasiness	0.60 (+ 0.28)	0.42 (- 0.04)	0.51 (+ 0.12)
Beauty/Joy	0.62 (- 0.13)	0.61 (- 0.14)	0.61 (- 0.13)
Vitality	0.54 (- 0.01)	0.45 (- 0.08)	0.50 (- 0.04)
Humor	0.62 (+ 0.17)	0.57 (+ 0.39)	0.59 (+ 0.28)
F1-macro Score	0.47 (+ 0.08)	0.48 (+ 0.13)	0.47 (+ 0.11)

Table 5.6.: F1-score for the emotion classification BERT models with poetry fine-tuned BERT as starting model and meter classification as intermediate task.
0/1: model trained on split_0 and tested on split_1. **0/1**: model trained on split_1 and tested on split_0. Score differences compared to baseline BERT model are given in ( ).

Label	0/1	1/0	Average
Suspense	- 0.12	+ 0.01	- 0.05
Awe/Sublime	+ 0.07	- 0.21	- 0.07
Sadness	- 0.03	+ 0.03	+ 0.00
Annoyance	+ 0.00	+ 0.00	+ 0.00
Uneasiness	+ 0.01	- 0.10	- 0.04
Beauty/Joy	- 0.04	+ 0.01	- 0.02
Vitality	- 0.14	+ 0.09	- 0.03
Humor	+ 0.03	- 0.02	+ 0.01
F1-macro Score	- 0.03	- 0.02	- 0.03

(a) Fairy tales emotion classification as intermediate task.

Label	0/1	1/0	Average
Suspense	+ 0.21	+ 0.11	+ 0.16
Awe/Sublime	- 0.17	+ 0.02	+ 0.08
Sadness	- 0.05	- 0.03	- 0.04
Annoyance	+ 0.02	- 0.05	- 0.01
Uneasiness	+ 0.07	- 0.05	+ 0.01
Beauty/Joy	- 0.08	- 0.05	- 0.06
Vitality	- 0.05	+ 0.00	- 0.03
Humor	+ 0.06	- 0.02	+ 0.02
F1-macro Score	+ 0.00	- 0.01	+ 0.00

(b) Meter classification as intermediate task

Table 5.7.: F1-score deltas between different starting models in intermediate task training.
Delta = (Score when using BERT-large as starting model) - (Score when using poetry fine-tuned BERT as starting model).

why this is the case, although we suspect that some of the helpful parameters that result from the fine-tuned BERT model are forgotten during the intermediate training phase, which leads to results that are somewhat identical to training with pre-trained BERT-large. We refer to Pruksachatkun et al. (2020) for a more in-depth discussion of intermediate task training, which explain the mechanics of intermediate task training and its benefits as a whole.

5.1.5. Conclusion

We perform three transfer learning experiments with fine-tuning BERT as the core principle. Our first experiment fine-tunes BERT-large with poetry data, while our second and third experiment incorporate intermediate task training before the final classification task. Each experiment uses a different dataset and a different task. Compared to the BERT baseline models, all 3 approaches show varying performance increases, with the poetry fine-tuning experiment being the most successful one (Figure 5.2). We hypothesize that introducing poetry data to BERT helps improving classification results the most, as seen by the two experiments with poetry-related datasets having better performance increases compared to the one with dataset from another literature source. As shown in Figure 5.3, on a per-label basis, the 3 experiments show improvements in general, especially on labels that are underrepresented in the PO-EMO dataset. In fact, two of them were able to make meaningful predictions for the *Annoyance* label, which the baseline BERT models are not able to do. Our results for the transfer learning experiments share the same observation by Phang, Févry, and Bowman (2019), Devlin et al. (2019), Pruksachatkun et al. (2020), etc that transfer learning with BERT can help training better models for classification tasks, especially in low-resource setting. Further experiments that combine fine-tuning on unlabeled data with intermediate task training show no improvement, which is quite surprising.

5.2. Data Augmentation

Data augmentation is the process of introducing new samples to the training dataset to increase the amount of data available in low-resource setting. New samples can be obtained by performing transformation techniques on the original data. These techniques include duplicating existing samples (Chawla et al. (2002)), replacing keywords (Kobayashi (2018), Xie et al. (2019)), creating similar samples (Xie et al. (2019), Sennrich, Haddow, and Birch (2016)), etc. By creating new samples in a unsupervised manner, one could augment as much data as needed to bypass the low-resource problems. However the quality of augmented data is critical, since it could negatively affect the performance of models by introducing noise or wrong signals during the training phase. As poetry has complicated aesthetics requirements and constraints, it is important to retain such qualities when augmenting poetry data. For our experiments, four different techniques were taken into consideration: *Oversampling, Back-translation, Replacing words* and *Stanza shuffling*.

5.2.1. Oversampling

Oversampling is the technique in which we add duplicated samples to training splits. By doing this, we can improve the performance on less populated labels, thus increase the overall result. On the other hand, as the samples added are exact copies, adding already well-represented labels could make the inherent imbalance between the number of samples for each label in our dataset worse. As our classification task is multi-label in nature, it is important that we only duplicate samples that only have the under-represented labels.

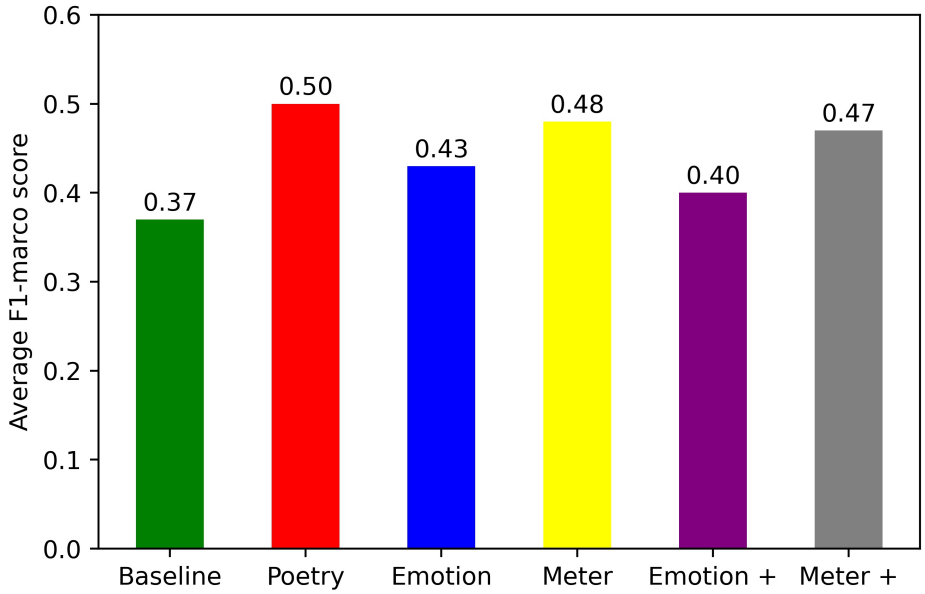


Figure 5.2.: Transfer learning: Average F1-macro scores.
Poetry: Poetry fine-tuning. **Emotion**: Emotion classification inter. task training. **Meter**: Meter classification inter. task training. **Emotion +**: Emotion classification inter. task training with poetry fine-tuned starting model. **Meter +**: Meter classification inter. task training with poetry fine-tuned starting model.

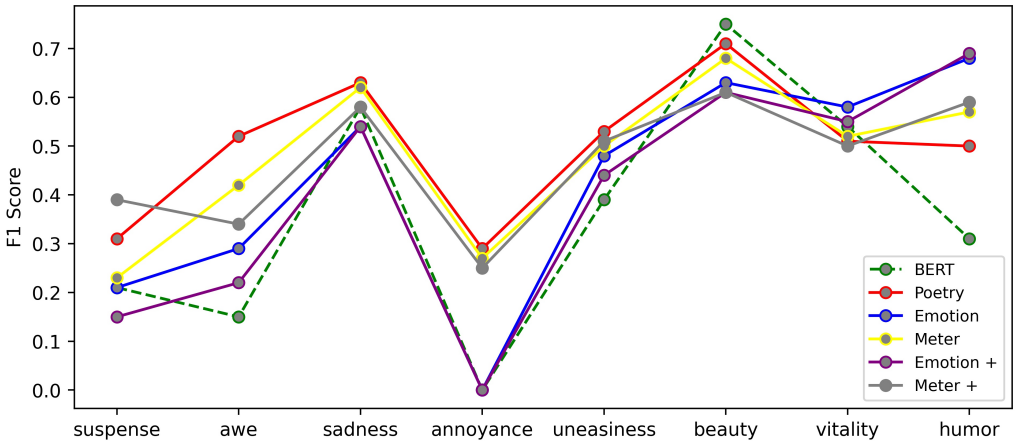


Figure 5.3.: Transfer learning: F1 scores per label.
Poetry: Poetry fine-tuning. **Emotion**: Emotion classification inter. task training. **Meter**: Meter classification inter. task training. **Emotion +**: Emotion classification inter. task training with poetry fine-tuned starting model. **Meter +**: Meter classification inter. task training with poetry fine-tuned starting model.

We categorize the labels into two subsets : *underpopulated* labels and *overpopulated* labels. Only samples from labels that are currently underpopulated in the training split are duplicated. To determine whether a label is underpopulated or not, we calculate the average sample counts for all labels. If a label has fewer samples than this average, we should duplicate samples that belong to it. We try to select samples that are only annotated with underpopulated labels whenever possible. If there is no sample that fulfills this condition, i.e samples with this underpopulated label all have overpopulated labels attached to it as well, then we have no choice but to use them as well. By using exact copies of stanzas annotated by experts, we maintain the overall quality of the dataset. Table 5.8 shows the number of samples for each label after oversampling for split_0 and split_1.

We train the models with oversampled training splits, but leave the test splits and BERT's classification layer untouched. The results for this experiment can be seen in Table 5.9.

Label	split_0 before	split_0 after	split_1 before	split_1 after
Suspense	13	31	11	34
Awe/Sublime	13	39	17	21
Sadness	34	34	33	37
Annoyance	5	20	6	21
Uneasiness	14	17	20	22
Beauty/Joy	39	39	39	40
Vitality	22	22	28	28
Humor	10	25	7	30

Table 5.8.: Sample counts for each label after oversampling. Underpopulated labels are marked with **bold**.

Label	0/1	1/0	Average
Suspense	0.17 (- 0.26)	0.47 (+ 0.47)	0.32 (+ 0.10)
Awe/Sublime	0.11 (+ 0.11)	0.00 (- 0.30)	0.06 (- 0.10)
Sadness	0.60 (+ 0.01)	0.66 (+ 0.09)	0.63 (+ 0.05)
Annoyance	0.00 (+ 0.00)	0.00 (+ 0.00)	0.00 (+ 0.00)
Uneasiness	0.26 (- 0.06)	0.50 (+ 0.05)	0.38 (- 0.01)
Beauty/Joy	0.73 (- 0.01)	0.65 (- 0.10)	0.69 (- 0.06)
Vitality	0.47 (- 0.09)	0.56 (+ 0.03)	0.51 (- 0.03)
Humor	0.40 (- 0.04)	0.57 (+ 0.39)	0.49 (+ 0.17)
F1-macro Score	0.34 (- 0.04)	0.43 (+ 0.08)	0.38 (+0.02)

Table 5.9.: F1-score for the emotion classification BERT models with oversampled training set. **0/1**: model trained on split_0 and tested on split_1. **0/1**: model trained on split_1 and tested on split_0. Score differences compared to baseline BERT model are given in ( ).

Our oversampling experiment seems to produce differing results: The macro F1-score for testing on split_1 decreases by 0.04, while the score for testing on split_0 increase by 0.08. Among the underpopulated labels, which this experiment is designed to improve performance on, only *Suspense* and *Humor* show increases in F1-scores, while the rest stays the same or gets slightly worse. The average F1-macro only increases by 0.01, which suggests oversampling might not be very helpful for our classification task.

5.2.2. Back-translation

Our second data augmentation experiment is an implementation of the idea by Xie et al. (2019). This idea relies on automatic translation systems to create new samples, using a two-step setting. First, existing samples are translated into a *pivot language*. After that, the translated samples are translated back from the pivot language to the original language. Compared to the original samples, the newly acquired ones have some differences in sentence structure or word usage, but the overall meaning and associated emotion labels remains unchanged. We translate an entire stanza, instead of doing it line-by-line to make sure the context remain consistent. However, as poetry is written with formal and aesthetic constraints, translating them back and forth might break the poems' forms. We choose French as our pivot language as suggested by the original paper. The translations in both directions are provided by Google Translate[4]. Table 5.10 is an example of a back-translated stanza.

Original text	Back-translated text
Music , when **soft** voices die , Vibrates in **the** memory — **Odours** , when sweet violets **sicken** , Live **within** the sense they **quicken** .	music when **sweet** voices die vibrates in memory - **smells** when sweet violets **get sick** live **in** the sense that they **accelerate**

Table 5.10.: Example: Back-translation. Changes are marked in **bold**.

For each sample in PO-EMO, we generate a new sample by back-translation. This effectively increases the amount of training data we have by 100%. By changing the pivot language, we can generate as many new samples as needed. However, due to time constraints, we are only able to test with one pivot language. We apply back-translation to the training splits and train models using the same BERT setup. The result of this experiment is in Table 5.11.

Label	0/1	1/0	Average
Size increase (%)	100	100	100
Suspense	0.40 (- 0.03)	0.25 (+ 0.25)	0.33 (+ 0.11)
Awe/Sublime	0.42 (+ 0.42)	0.38 (+ 0.08)	0.40 (+ 0.25)
Sadness	0.57 (- 0.02)	0.67 (+ 0.10)	0.62 (+ 0.04)
Annoyance	0.00 (+ 0.00)	0.25 (+ 0.25)	0.13 (+ 0.13)
Uneasiness	0.56 (+ 0.24)	0.50 (+ 0.05)	0.53 (+ 0.14)
Beauty/Joy	0.72 (- 0.02)	0.44 (- 0.31)	0.58 (- 0.17)
Vitality	0.68 (+ 0.12)	0.41 (- 0.12)	0.54 (+ 0.00)
Humor	0.67 (+ 0.22)	0.46 (+ 0.28)	0.56 (+ 0.25)
F1-macro Score	0.50 (+ 0.12)	0.42 (+ 0.07)	0.46 (+ 0.09)

Table 5.11.: F1-score for the emotion classification BERT models with back-translated training set. **0/1**: model trained on split_0 and tested on split_1. **0/1**: model trained on split_1 and tested on split_0. Score differences compared to baseline BERT model are given in ( ).

In this experiment, we double the amount of training data for all labels, instead of just the underpopulated ones. This seems to have an overall positive impact on the overall results, as 6 out of 8 labels have better average F1-scores. However, only one of the models can make meaningful predictions for the *Annoyance* label, which suggests that back-translation might not be helpful enough to help us overcome this particular problem.

[4] `https://pypi.org/project/googletrans/`

5.2.3. Words Replacement

In our third data augmentation experiment, we generate new samples by replacing unimportant words in existing examples with similar words. The idea is inspired by researches by Kobayashi (2018) and Xie et al. (2019). The goal of this is also to create new samples without changing the annotated labels of the stanzas.

We identify the unimportant words by using tf-idf score. The idea is that if a words appear multiple times across multiple documents in a corpus (for example, stopwords), it might not be very meaningful or might not carry any important context. On the other hand, if a word is rarely used in a corpus and appears many times in a single document, it could contain valuable information to the content of that document. Tf-idf reflects this by multiplying the *term frequency* of a word (the number of times it appears in a single document), with its *inverted document frequency* (the logarithmic inverted number of documents this word has appeared in within a collection of documents). The lower the tf-idf of a word is, the less important it is. In our experiment, we choose the term frequency as the number of times a word appears in a stanza, and the inverted document frequency as the number of times a word appears in the entire split that has the stanza. For each sample, we calculate the tf-idf[5] score for every word and choose the least important word from each line to replace. Table 5.12 shows a stanza that is fully word-replaced.

We choose a suitable word for replacement by looking at a word embeddings space (Mikolov et al. (2013)). We use a pre-trained gensim Word2Vec model [6] for the embeddings. The model has a large vocabulary and covers many forms of literature, as it is trained on the GoogleNews data set for 3 million words and phrases. For each unimportant word, we find the top three most similar one from the embeddings space and randomly take one of them as the replacement. After replacing the least important words from each line, we get a new stanza. We make a new stanza out of each existing sample. As with other experiments, we apply the technique to the training set only and report the result in Table 5.13.

Original text	Replaced text
Music , when soft voices **die** ,	music, when soft voices **perish**
Vibrates **in** the memory —	vibrates **inthe** the memory —
Odours , when **sweet** violets sicken ,	odours when **sweetness** violets sicken
Live within **the** sense they quicken .	live within **this** sense they quicken

Table 5.12.: Example: Word replacement. Changes are marked in **bold**.

The result for word replacement experiment is rather surprising, as our models perform better on all under-populated labels. Not only that, the model trained on split_0 and its word-replaced samples have a large increase in performance. The model trained on augmented split_1 also has better score, but not as noticeable. Compared to the back-translation experiment (Section 5.2.2) where the same amount of extra training data is added, making minimal changes instead of changing large parts of the existing stanzas yields better results. It could also be a consequence of how we systematically change the features, i.e using a measure to identify the replaceable words and replace them with similar words from an embeddings space, rather than relying on potentially unreliable translations. Word replacement also keeps the poetry form relatively intact, which machine translation might not be able to do.

[5] Calculated with `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`

[6] `https://radimrehurek.com/gensim/auto_examples/tutorials/run_word2vec.html`

Label	0/1	1/0	Average
Size increase (%)	100	100	100
Suspense	0.42 (- 0.01)	0.47 (+ 0.47)	0.45 (+ 0.23)
Awe/Sublime	0.45 (+0.45)	0.21 (- 0.09)	0.33 (+ 0.18)
Sadness	0.67 (+ 0.08)	0.61 (+ 0.04)	0.64 (+ 0.06)
Annoyance	0.29 (+ 0.29)	0.00 (+ 0.00)	0.14 (+ 0.14)
Uneasiness	0.43 (+ 0.11)	0.57 (+ 0.12)	0.50 (+ 0.11)
Beauty/Joy	0.64 (- 0.11)	0.62 (- 0.12)	0.63 (- 0.12)
Vitality	0.60 (+ 0.05)	0.49 (- 0.04)	0.54 (+ 0.00)
Humor	0.67 (+ 0.22)	0.57 (+ 0.39)	0.62 (+ 0.31)
F1-macro Score	0.52 (+ 0.14)	0.44 (+ 0.10)	0.48 (+ 0.11)

Table 5.13.: F1-score for the emotion classification BERT models with training set enriched by replacing words.
0/1: model trained on split_0 and tested on split_1. **0/1**: model trained on split_1 and tested on split_0. Score differences compared to baseline BERT model are given in ( ).

5.2.4. Stanza Shuffling

Our last data augmentation experiment is stanza shuffling, in which we randomly shuffle the lines in a stanza to generate new samples. This approach is based on how we model our classification task, which we look at emotions on a stanza-level. Stanza shuffling allows us to introduce relatively little noise to the dataset since no changes are made to the sentence structure or the wording of lines, but the ordering of the lines and their dependencies are changed. The labels annotated to each stanza also remain unchanged. As with back-translation and word replacement, we make a new stanza out of an existing one and train with the same classifier layer as the baseline BERT models. The results are in Table 5.14.

Label	0/1	1/0	Average
Size increase (%)	100	100	100
Suspense	0.29 (- 0.14)	0.35 (+ 0.35)	0.32 (+ 0.11)
Awe/Sublime	0.27 (+ 0.27)	0.44 (+ 0.14)	0.35 (+ 0.20)
Sadness	0.62 (+ 0.03)	0.70 (+ 0.13)	0.66 (+ 0.08)
Annoyance	0.00 (+ 0.00)	0.00 (+ 0.00)	0.00 (+ 0.00)
Uneasiness	0.44 (+ 0.12)	0.63 (+ 0.17)	0.53 (+ 0.15)
Beauty/Joy	0.68 (- 0.06)	0.71 (- 0.04)	0.70 (- 0.05)
Vitality	0.64 (+ 0.09)	0.55 (+ 0.02)	0.59 (+ 0.05)
Humor	0.67 (+ 0.22)	0.46 (+ 0.28)	0.56 (+ 0.25)
F1-macro Score	0.45 (+ 0.07)	0.48 (+ 0.13)	0.46 (+ 0.09)

Table 5.14.: F1-score for the emotion classification BERT models with training set enriched by stanza replacing.
0/1: model trained on split_0 and tested on split_1. **0/1**: model trained on split_1 and tested on split_0. Score differences compared to baseline BERT model are given in (.)

The experiment shows good overall improvements, although as with many other approaches some labels are still not classified properly. This, however, opens the question as to how important it is to take the lines' order into consideration when we classify emotion on a stanza level, as adding scrambled lines has a positive effect

on the overall result.

5.2.5. Conclusion

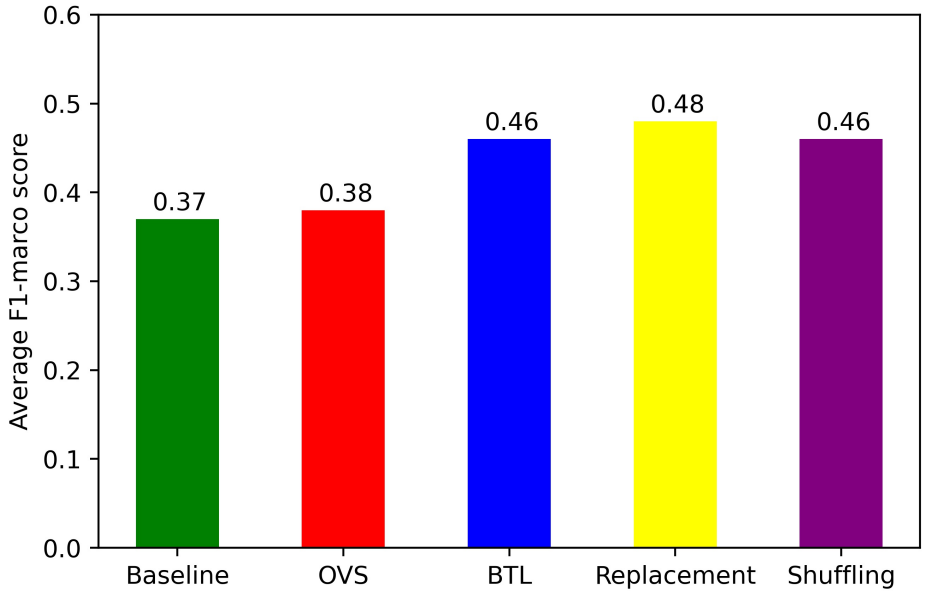


Figure 5.4.: Data augmentation: Average F1-macro scores.
OVS: Oversampling. **BTL**: Back-translation. **Replacement**: Word replacement. **Shuffling**: Stanza shuffling.

Figure 5.5.: Data augmentation: F1 scores per label.

We research 4 methods of data augmentation to improve our baseline models. While there are improvements

in the macro F1-score for all of them (see Figure 5.4), it's not as noticeable as the transfer learning experiments. As shown in Figure 5.5, most of them are still incapable of dealing with the label *Annoyance*, which is one of the main problems with the baseline models. This could be attributed to the lack of samples with *Annoyance* in the training set (Table 3.4). Despite that, we still have some improvements across all labels, which is encouraging. Data augmentation is an inexpensive and straight-forward way to deal with low amount of training data, although it is not without drawbacks: mainly a lack of quality by having to compromise certain aspects (line structures, word usage, etc) of the original data. This is important to take into consideration in the context of our classification task, as poetry is very much bounded to stylistic constraints.

5.3. Cross-lingual Learning

Cross-lingual learning refers to the usage of data that is available in different languages to help improving performance on a language with fewer resource. As mentioned in Chapter 3.1, the PO-EMO dataset that we use in our experiments includes poems written in both German and English. There are more German poems than English poems in this dataset, as reported in Chapter 3. Since the poems in both languages are annotated by the same annotators, with the same set of labels and under the same criteria, the quality of the data should not be a problem. We perform three cross-lingual experiments with the data from PO-EMO.

5.3.1. Baseline Models with BERT-multilingual

Since our baseline models are based on BERT-large, which does not have multi-lingual support, we decide to use a different variant of BERT instead. This variant of BERT is called multi-lingual BERT (m-BERT). Since m-BERT has different parameters and configurations compared to BERT-large[7], it makes sense to train new baseline models using m-BERT for better comparison. Using the exact setting described in Chapter 4.2 for the classifier layer, we train two new baseline models with the English data only and receive the results in Table 5.15.

Label	0/1	1/0	Average
Suspense	0.14 (- 0.29)	0.38 (+ 0.38)	0.26 (+ 0.04)
Awe/Sublime	0.00 (+ 0.00)	0.30 (+ 0.00)	0.15 (+ 0.00)
Sadness	0.54 (- 0.05)	0.21 (- 0.36)	0.37 (- 0.21)
Annoyance	0.00 (+ 0.00)	0.00 (+ 0.00)	0.00 (+ 0.00)
Uneasiness	0.26 (- 0.06)	0.44 (- 0.01)	0.35 (- 0.04)
Beauty/Joy	0.62 (- 0.13)	0.65 (- 0.09)	0.64 (- 0.11)
Vitality	0.60 (+ 0.04)	0.53 (+ 0.00)	0.56 (+ 0.02)
Humor	0.82 (+ 0.38)	0.43 (+ 0.25)	0.63 (+ 0.31)
F1-macro Score	0.37 (- 0.01)	0.37 (+ 0.01)	0.37 (+ 0.00)

Table 5.15.: F1-score for the baseline m-BERT models.
0/1: model trained on split_0 and tested on split_1. **0/1**: model trained on split_1 and tested on split_0. Score differences compared to baseline BERT model are given in ( ).

The baseline models for m-BERT have the same average F1-macro score compared to the baseline BERT-large model. Out of the 8 emotion labels, *Sadness* and *Humor* show noticeable differences, with *Sadness* has worse

[7]BERT-large: 24-layer, 1024-hidden, 16-heads, 335M parameters. m-BERT: 12-layer, 768-hidden, 12-heads, 179M parameters

score and *Humor* has better score compared to BERT-large. The rest of the labels has little or no deviation in score. Overall, since changing the base structure doesn't seem to affect our baseline results, we can compare our results for the cross-lingual experiments below with other experiments without any difficulty.

5.3.2. Using German data as Additional Training Data

In this experiment, we directly add the German data to the training sets. In both scenarios (training with either split_0 or split_1), there are 512 German stanzas and 87 English stanzas available for training. The German stanzas are represented in the same way as English ones: We concatenate all lines in one stanza into one line with special tokens to separate lines. We then train new models with the enriched training splits, keep the testing splits the same and report the results in Table 5.16).

Label	0/1	1/0	Average
Suspense	0.25 (+ 0.11)	0.12 (- 0.26)	0.18 (- 0.08)
Awe/Sublime	0.35 (+ 0.35)	0.34 (+ 0.04)	0.35 (+ 0.19)
Sadness	0.55 (+ 0.02)	0.54 (+ 0.34)	0.55 (+ 0.18)
Annoyance	0.17 (+ 0.17)	0.33 (+ 0.33)	0.25 (+ 0.25)
Uneasiness	0.53 (+ 0.27)	0.58 (+ 0.14)	0.56 (+ 0.21)
Beauty/Joy	0.61 (- 0.01)	0.64 (- 0.01)	0.63 (- 0.01)
Vitality	0.51 (- 0.08)	0.37 (- 0.16)	0.44 (- 0.12)
Humor	0.43 (- 0.40)	0.18 (- 0.25)	0.31 (- 0.32)
F1-macro Score	0.43 (+ 0.05)	0.39 (+ 0.02)	0.41 (+ 0.01)

Table 5.16.: F1-score for the m-BERT models with additional German training data. **0/1**: model trained on split_0 and tested on split_1. **0/1**: model trained on split_1 and tested on split_0. Score differences compared to baseline m-BERT model are given in ( ).

We observe some moderate improvements over the baseline models, but not as much as we expect. One positive result is that our models manage to classify *Annoyance* in this experiment, which many approaches before fail to classify. Although there is only a total of 25 German samples with this label, including all of them in the training stage seems to be helpful. Our data augmentation approaches (Chapter5.2) also increase this label's samples by a comparable amount of new samples and have mixed results when trying to classify it. This could be a result of the PO-EMO samples are authentic stanzas and annotated by experts.

5.3.3. Oversampling with m-BERT

We combine a data augmentation technique with cross-lingual learning in this experiment. We choose Oversampling (Section 5.2.1) as the technique. We oversample the English data only, and train the model together with German data.

Table 5.17 shows the results for this experiment, which are still not good enough. Introducing duplicated English data seems to have an negative impact on classifying *Annoyance*, as one of the models fail to classify it again.

Label	0/1	1/0	Average
Suspense	0.15 (+ 0.01)	0.35 (- 0.02)	0.25 (- 0.01)
Awe/Sublime	0.48 (+ 0.48)	0.29 (- 0.02)	0.38 (+ 0.23)
Sadness	0.59 (+ 0.06)	0.65 (+ 0.45)	0.62 (+ 0.25)
Annoyance	0.13 (+ 0.13)	0.00 (+ 0.00)	0.06 (+ 0.06)
Uneasiness	0.63 (+ 0.36)	0.62 (+ 0.17)	0.62 (+ 0.27)
Beauty/Joy	0.43 (- 0.18)	0.61 (- 0.04)	0.52 (- 0.11)
Vitality	0.54 (- 0.06)	0.44 (- 0.09)	0.49 (- 0.07)
Humor	0.46 (- 0.36)	0.33 (- 0.10)	0.40 (- 0.23)
F1-macro Score	0.43 (+ 0.05)	0.41 (+ 0.04)	0.42 (+ 0.05)

Table 5.17.: F1-score for the m-BERT models with additional German training data and oversampled English data.
0/1: model trained on split_0 and tested on split_1. **0/1**: model trained on split_1 and tested on split_0. Score differences compared to baseline m-BERT model are given in ( ).

5.3.4. Fine-tuning m-BERT with German Data

We apply fine-tuning with m-BERT here to make use of the German data from PO-EMO. For the fine-tune dataset, we only use the German stanzas without the label, and we use the Masked Language Modeling task as the fine-tune task. After the fine-tuning step, we train the classification model on the two splits for multi-label classification and get the following results in Table 5.18.

Label	0/1	1/0	Average
Suspense	0.40 (+ 0.26)	0.38 (+ 0.00)	0.39 (+ 0.13)
Awe/Sublime	0.33 (+ 0.33)	0.19 (- 0.11)	0.26 (+ 0.11)
Sadness	0.58 (+ 0.04)	0.59 (+ 0.38)	0.58 (+ 0.21)
Annoyance	0.00 (+ 0.00)	0.33 (+ 0.33)	0.17 (+ 0.17)
Uneasiness	0.44 (+ 0.18)	0.38 (- 0.06)	0.41 (+ 0.06)
Beauty/Joy	0.60 (- 0.02)	0.52 (- 0.13)	0.56 (- 0.08)
Vitality	0.47 (- 0.13)	0.42 (- 0.11)	0.44 (- 0.12)
Humor	0.71 (- 0.11)	0.43 (+ 0.00)	0.57 (- 0.05)
F1-macro Score	0.44 (+ 0.07)	0.40 (+ 0.04)	0.42 (+ 0.05)

Table 5.18.: F1-score for the emotion classification BERT models based on poetry fine-tuned m-BERT.
0/1: model trained on split_0 and tested on split_1. **0/1**: model trained on split_1 and tested on split_0. Score differences compared to baseline m-BERT model are given in ( ).

Compared to Section 5.1.1 which fine-tunes on BERT-large with 30,000 stanzas before the emotion classification task, fine-tuning m-BERT doesn't have quite the same improvement. However, this could be due to the large difference between the amount of fine-tune data uses (30,000 stanzas and 512 stanzas, respectively). As pointed out in Section 5.1.5, the increase in classifying performance can be attributed to using poetry-related dataset in a transfer learning setup.

5.3.5. Intermediate Task Training with German Poetry Emotion Classification

In our third experiment, we apply intermediate task training with a multi-label classification task using the annotated German data as the intermediate task. From the 512 German stanzas, we randomly selected 409 stanzas (80% of the available data) for training and use the rest for testing. The labels for this classification are the same as the English one, as the data comes from the same dataset. The results for German classification task is in Table 5.19a. We then use the resulting model to train the English multi-label classification task and report the result in Table 5.19b.

Label	F1-Score
Suspense	0.24
Awe/Sublime	0.10
Sadness	0.40
Annoyance	0.29
Uneasiness	0.45
Beauty/Joy	0.60
Vitality	0.23
Humor	0.10
F1-macro Score	0.30

(a) F1-score for multi-label emotion classification with German data. Trained on 409 stanzas, tested on 103 stanzas

Label	0/1	1/0	Average
Suspense	0.43 (+ 0.29)	0.14 (- 0.23)	0.29 (+ 0.03)
Awe/Sublime	0.35 (+ 0.35)	0.47 (+ 0.16)	0.41 (+ 0.26)
Sadness	0.51 (-0.03)	0.43 (+ 0.22)	0.47 (+ 0.10)
Annoyance	0.29 (+ 0.29)	0.25 (+ 0.25)	0.27 (+ 0.27)
Uneasiness	0.61 (+ 0.35)	0.62 (+ 0.17)	0.61 (+ 0.26)
Beauty/Joy	0.64 (+ 0.02)	0.58 (- 0.07)	0.61 (- 0.03)
Vitality	0.48 (- 0.12)	0.49 (- 0.04)	0.48 (- 0.08)
Humor	0.29 (- 0.54)	0.33 (- 0.10)	0.31 (-0.32)
F1-macro Score	0.45 (+ 0.08)	0.41 (+ 0.05)	0.43 (+ 0.06)

(b) F1-score for the emotion classification BERT models with emotion classification on German poems as intermediate task. **0/1**: model trained on split_0 and tested on split_1. **0/1**: model trained on split_1 and tested on split_0. Score differences compared to baseline m-BERT model are given in ( ) .

Table 5.19.: Result: Intermediate task training with German poetry emotion classification task

For the German classification task, our result is bad. Haider et al. (2020) also use m-BERT to classify the German stanzas (albeit with a slightly different setup, particularly in the distribution of training/testing data) and receives a F1-micro of 0.34 on the test set. Their best experiment with the German poems uses German-specialized BERT model and have a 0.52 F1-micro score. This suggests that m-BERT might not be very effective in mono-lingual tasks compared to BERT models that are pre-trained specifically for the target language. Nevertheless, our English classification task does have the most improvements among the cross-lingual experiments so far, with an average F1-macro score of 0.43.

5.3.6. Conclusion

We try out four cross-lingual approaches to improve performance in low-resource settings. The approaches manage to accomplish this goal, although the improvements are not very good in term of macro F1-score (Figure 5.6). However, on a per-label basis (Figure 5.7), there are good improvements on underpopulated labels in the dataset, such as *Awe/Sublime*, *Uneasiness* and most importantly, *Annoyance*. This could be attributed to the high quality additional data with those labels that is available in a different language. The baseline experiment for m-BERT also shows that with a small dataset, the choice of pre-trained BERT models makes no impact on the results, although this might not be the case for larger dataset.

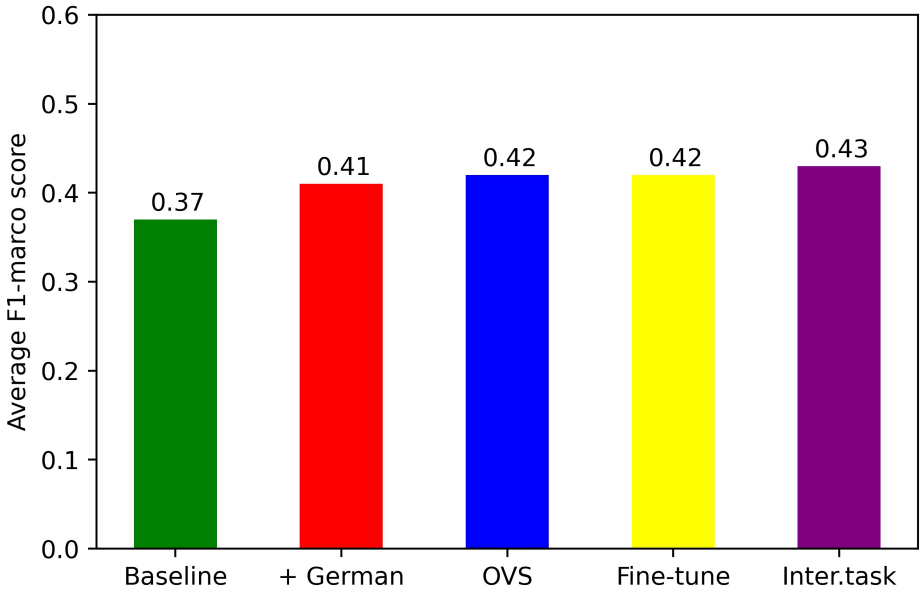


Figure 5.6.: Cross-lingual: Average F1-macro scores.
+ German: German stanzas as additional training data. **OVS**: German stanzas as additional training data, oversampled English data. **Fine-tune**: Fine-tune with German stanzas. **Inter.task**: Intermediate task training with German stanzas classification.

Figure 5.7.: Cross-lingual: F1 scores per label.
+ German: German stanzas as additional training data. **OVS**: German stanzas as additional training data, oversampled English data. **Fine-tune**: Fine-tune with German stanzas. **Inter.task**: Intermediate task training with German stanzas classification.

5.4. Ensemble

In previous Sections, we perform various experiments to increase classification performance in low-resource setting. Individually, the results from the experiments are quite encouraging, as they show varying degrees of improvement compared to the baseline models. However, we fail to further improve on this, as our attempts to combine these experiments (Section 5.1.4, 5.3.3, 5.3.4, 5.3.5) all have no satisfying outcomes. As mentioned in Section 2.3.4, combining classifiers in an ensemble setting can improve the overall results and get a close approximation to the optimal solution for our classification task. Therefore, we perform ensemble on the output of our classifiers (or models) in our last experiment.

We follow the framework by Nguyen et al. (2020) to combine the outputs of our classifiers. We define the following operation for the predict-then-combine (PTC) and combine-then-predict (CTP) strategies.

- **CTP**

  Intersection Get the binary predictions from the classifiers and perform INTERSECTION operation on the predictions to make the final prediction.

  Union Get the binary predictions from the classifiers and perform UNION operation on the predictions to make the final prediction.

  K-label voting If more than k classifiers agree that a label is relevant for a prediction, that label is considered correct for the final prediction.

- **PTC**

  Average Get the probabilities predictions from the models on their appropriate test split, calculate the average probabilities and apply a threshold $\sqcup = 0.5$ on the averaged probabilities.

As a recap, here are the classifiers that we will use, our naming convention which we use to report the results with and the sections they are described in. From our 10 experiments, there exist 10 classifiers that are trained on split_0, and 10 classifiers that are trained on split_1 for a total of 20 classifiers. Only classifiers which are trained on the same splits are included in an ensemble. We measure the performance of an ensemble by measuring the F1 score of the predictions it makes on its test split, i.e the one that the member classifiers are **not** trained on.

poetry_pretraining Transfer learning with fine-tuning poetry (Section 5.1.1).

emotion_finetuning Transfer learning with emotion classification in fairy tales as upstream task (Section 5.1.2).

meter_finetuning Transfer learning with meter classification in poetry as upstream task (Section 5.1.3).

oversampling Oversampling of underpopulated labels (Section 5.2.1).

back_translation Back-translation (Section 5.2.2).

word_replacing Word replacement (Section 5.2.3).

stanza_shuffling Stanza shuffling (Section 5.2.4).

m-BERT_de m-BERT trained with additional German data (Section 5.3.2).

m-BERT_finetune m-BERT fine-tuned with poetry (Section 5.3.4).

m-BERT_finetune_cl m-BERT fine-tuned with emotion classification in German poetry as upstream task (Section 5.3.5).

For example, we want to combine the outputs of poetry_pretraining with emotion_finetuning. There are 2 ensembles that can be built accordingly, (poetry_pretraining, emotion_finetuning)_01 and (poetry_pretraining, emotion_finetuning)_10[8]. We measure the F1-macro scores for them, and the overall performance is the average of these two F1-macro scores.

5.4.1. Ensembles of Pairs and Triples of Models

For the first experiment, we build ensembles from pairs of models and from triples of models. For brevity, we refer to the ensembles that are built from pairs of models as 2-ensembles and ones that are built from triples of models as 3-ensembles in further discussion. We use the following operations to combine the outputs: **Intersection, Union** and **Average**. The full results of 2-ensembles can be found in Table B.1, B.2 and B.3. The partial results for 3-ensembles can be found in Table C.1, Table C.2 and Table C.3. We present the most relevant findings in Table 5.20.

	Average increase			Best		
Operation	INTER.	UNION	AVG	INTER.	UNION	AVG
2-ensemble	0.00	0.14	0.07	0.44	0.55	0.50
3-ensemble	-0.06	0.15	0.09	0.40	0.56	0.35

Table 5.20.: Relevant findings from 2-ensembles and 3-ensembles. **Average increase**: Average gains in F1-macro of all ensembles created with corresponding operation. **Best performance**: Average F1-macro across two splits of the best ensemble with corresponding operation. **INTER., UNION, AVG**: Intersection, union and average operation, respectively.

Union is by far the best ensemble operation for our task, as it is the only operation that have gains in both the average performance and the best possible performance in both 2-ensemble and 3-ensemble settings. By looking at the difference in scores between the Union and Intersection operations, we can say that our classifiers might not agree with each other when making predictions for certain samples. This is also observable in the their per-label performances reported before, as some of them fail to classify rare labels completely.

5.4.2. Ensemble of All Models

We combine the predictions from all classifiers in this experiment. In addition to the operations we use in Section 5.4.1, we also use **k-label voting**, where k can be 2, 3, 4 or 5 votes. We report the results for each operation in Table 5.21.

As expected, the Intersection operation performs the worst, since in this case the ensemble can only mark a label prediction as 'relevant' when *all* 10 classifiers agree on it, which is quite difficult. The Union operation also perform worse than in 2-ensemble and 3-ensemble settings, presumably because of the same reasoning: the ensemble has to mark a label prediction as 'relevant' if *any* of the classifiers does so, which makes it difficult to eliminate bad predictions from weak classifiers. K-label voting provides a good alternative to these

[8]The suffix **01** indicates trained on spilt_0, tested on split_1. The suffix **10** indicates trained on split_1, tested on split_0.

Operation	0/1	1/0	Average F1-macro
INTERSECTION	0.14 (- 0.24)	0.13 (- 0.22)	0.14 (- 0.23)
UNION	0.51 (+ 0.13)	0.47 (+ 0.12)	0.49 (+ 0.12)
k = 2	0.56 (+ 0.18)	0.52 (+ 0.17)	0.54 (+ 0.17)
k = 3	0.57 (+ 0.19)	0.50 (+ 0.15)	0.53 (+ 0.16)
k = 4	0.51 (+ 0.13)	0.48 (+ 0.13)	0.50 (+ 0.13)
k = 5	0.50 (+ 0.12)	0.51 (+ 0.16)	0.51 (+ 0.14)
AVERAGE	0.45 (+ 0.13)	0.48 (+ 0.13)	0.47 (+ 0.13)

Table 5.21.: F1-scores ensemble of all models. **Operation**: ensemble operation used. **0/1**: ensemble of models trained on split_0 and tested on split_1. **0/1**: ensemble of models trained on split_1 and tested on split_0. Score differences compared to baseline BERT model are given in ( ).

two extremes, as the ensemble needs a certain amount of members to agree on a prediction before accepting it. Our results show that k = 2 and k = 3 perform similarly well, while k = 4 and 5 perform a bit worse. The Average operation also perform worse in this experiment, which can also be attributed to the large number of classifiers involved.

5.4.3. Conclusion

For our last performance-enhancing experiment, we build ensembles of classifiers by combining their outputs. Despite the simplicity of the methods used, the ensembles perform well and improve on the individual experiments, which we couldn't accomplish before. We also find out the best overall model to use in our poetry analysis, which is the triple of (mbert_finetune_classification, stanza_shuffling, word_replacing) with average F1-score of 0.56 across two splits (Table C.2). With this, we conclude our experiments to increase performance for poetry emotion classification in small data setting.

6. Analysis

6.1. Prediction Analysis

After going through various experiments, we manage to build poetry emotion classification models that perform better than our baseline models. In order to get a deeper understanding of how and why our models are doing better, we perform a small model analysis.

For our analysis, we use LIME (short for **L**ocal **I**nterpretable **M**odel-agnostic **E**xplanations) by Ribeiro, Singh, and Guestrin (2016). LIME explains the prediction of a model for a given input sample by modifying the input and observe the predictions that model makes on the modified samples. In our case, where the input is a text, words are randomly removed from the input to make new samples. The model then makes predictions on these incomplete samples. LIME compiles the predictions and explains which words contribute the most to the prediction for the original input. An example of LIME's explanation can be seen in Figure 6.1.

We use LIME to gather the features (i.e words) that contribute the most to the predictions of our models. We do this by asking LIME to explain the predictions made by our models on all samples of their respective test sets. For each prediction, we look at the labels that are predicted as correct and take note of all features that positively contribute to them. After gathering this data from all predictions made by all models, we categorize them on a label basis, i.e listing which features are relevant to a label. We list the top 5 most relevant words for each label and their frequencies in Figure 6.2.

The LIME explanations give credibility to our models when predicting popular labels in the dataset. For example, when predicting the label *Beauty/Joy*, words such as **love, sweet** and **heart** are considered quite valuable, which in our opinion would be close to how a person would attribute this emotion label to poems or stanzas. The same observation can be made for *Awe/Sublime* and features such as **world** or **heaven**. An interesting result is that the word **death** and its variants (dead, dying, etc) contributes to 4 out of 8 labels (*Suspense, Sadness, Uneasiness* and *Humor*), most of which have negative connotation. Explanations for underpopulated labels such as *Annoyance* or *Humor* are somewhat unreliable, since there are not enough samples predicted with those labels to draw a meaningful conclusion. Overall, using LIME has given useful insight to how our models function, which can be further expanded by looking into, for example n-grams instead of just unigram frequency in future works. However, looking at words / features in isolation is certainly not representative of how emotions should be classified in poetry, and we acknowledge this shortcoming of our model analysis.

6.2. Poetry Analysis

We perform poetry analysis on a small, unlabeled set of data as an exploratory research to see how well our models can contribute to real-life applications. We decide to use our emotion classifiers on stanzas extracted

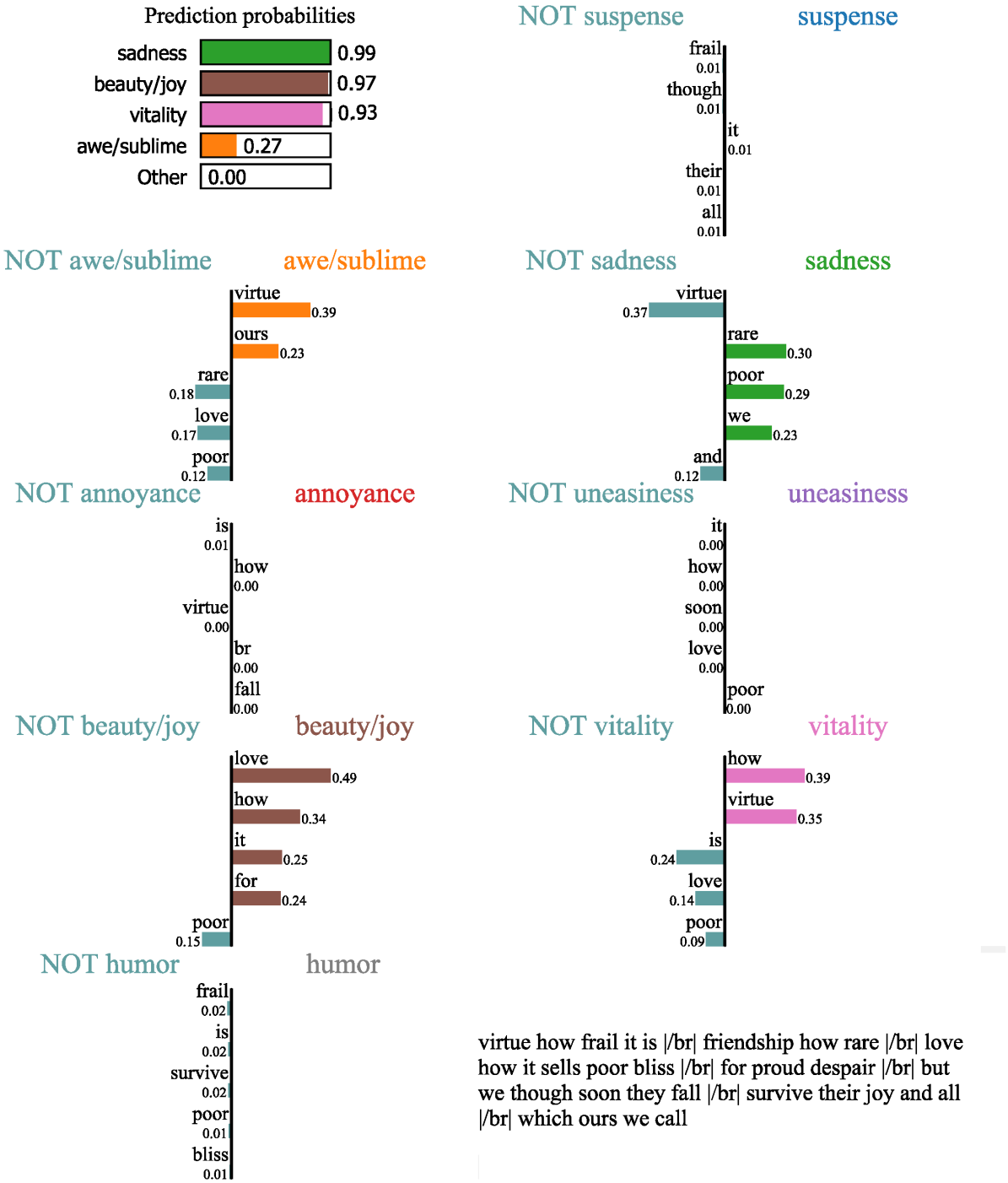
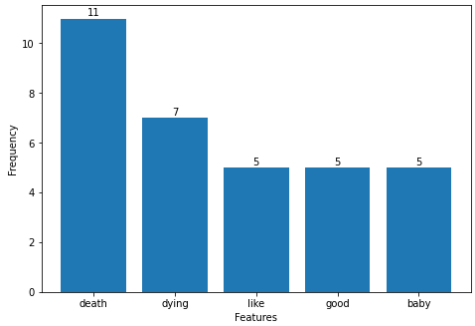
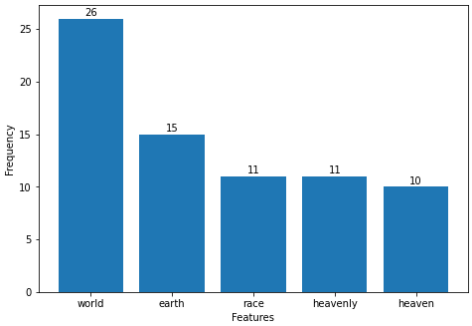


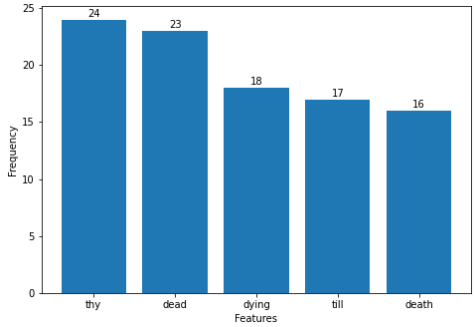
Figure 6.1.: A LIME explanation for a prediction on a stanza from our dataset. For each label, the most contributing words are shown. A positive value indicates a positive contribution to the prediction of this label, and negative value indicates a negative contribution.



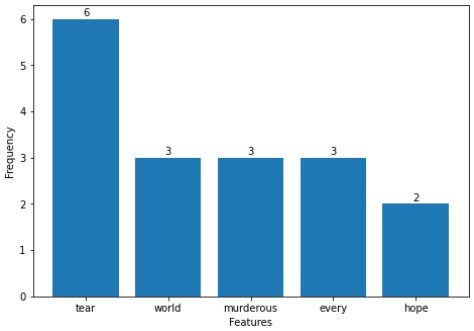
(a) Most common features for Suspense label



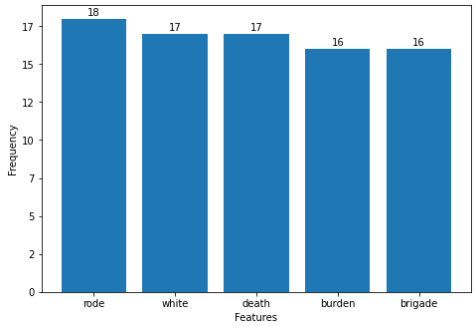
(b) Most common features for Awe/Sublime label



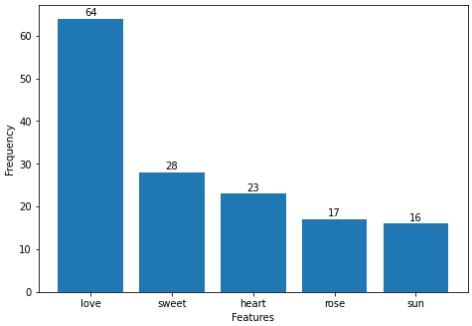
(c) Most common features for Sadness label



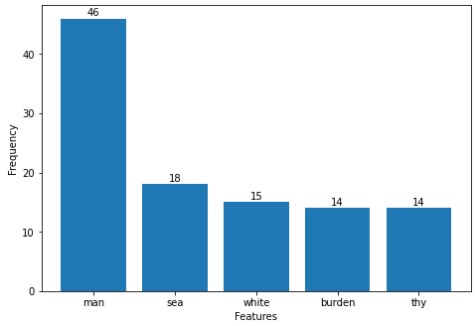
(d) Most common features for Annoyance label



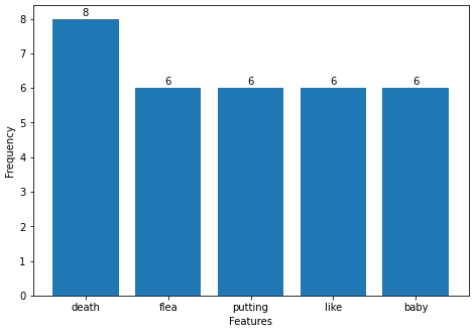
(e) Most common features for Uneasiness label



(f) Most common features for Beauty/Joy label



(g) Most common features for Vitality label



(h) Most common features for Humor label

Figure 6.2.: Top 5 most relevant features for each label by LIME.

from the Gutenberg English Poetry Corpus[1]. The corpus contains poems between the 16th and the 20th century. Poems are categorized in time periods, each time period lasts 100 years. The exact time period and number of stanzas can be seen in Table 6.1. From each time period, we randomly take 1000 stanzas for classification. For the 1900 - 2000 period, we take all available stanzas since the number of available stanzas is limited in this period. We use the best setup from our experiments, which is the ensemble of (mbert_finetune_classification, stanza_shuffling, word_replacing) (Section 5.4.3) to classify the stanzas. We present the results in Figure 6.3.

Time period	1500 - 1600	1600 - 1700	1700 - 1800	1800 - 1900	1900 - 2000
No. of available stanzas	5818	9728	47164	267495	943

Table 6.1.: Number of available stanzas from the Gutenberg English Poetry Corpus.

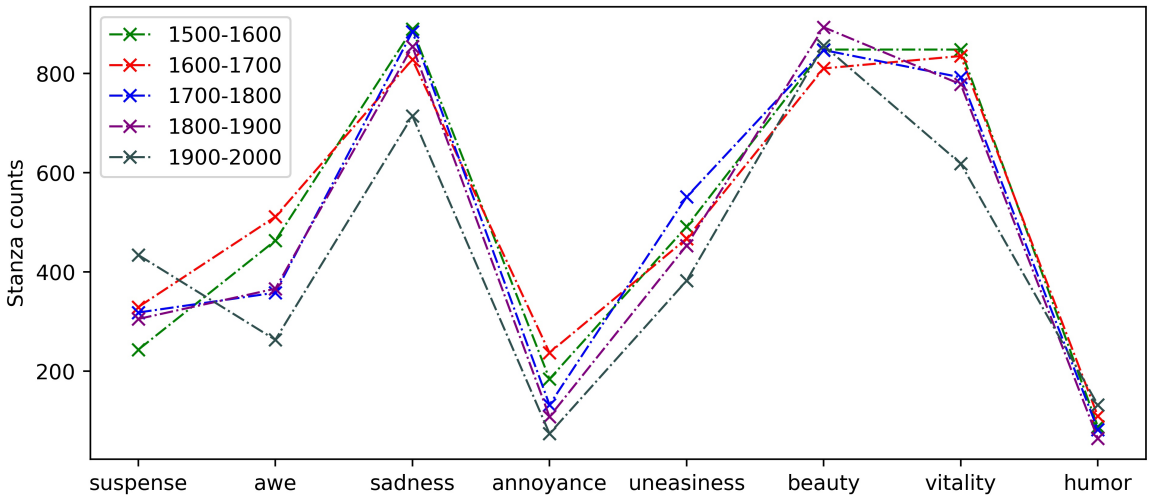


Figure 6.3.: Distribution of aesthetic emotions in stanzas from different time periods.

At first glance, the emotion distribution is quite similar for all time periods. In all of them, we can see that there are a lot of stanzas that have the labels *Sadness* and *Beauty/Joy*. There is a noticeable declining trend for *Awe/Sublime* starting from the 17th century, as there are much fewer stanzas with this label as time goes on. The same observation can be made for *Annoyance*, although as mentioned in Section 6.1 the predictions for this label might not be reliable. In the 20th century, we see a lot of more stanzas with *Suspense* and *Humor*. This is interesting to see since this period has fewer stanzas to classify on compared to other periods.

Overall, our poetry analysis doesn't show any conclusive results, as we only use a limited amount of data in this research. However, we show that there are some clear trends for emotions in poetry that might have potential for further research. We can also combine our current research in poetry emotions with other options; such as author profiling, stylistic profiling, etc to get a better understanding of poetry from a NLP perspective.

[1] `https://github.com/tnhaider/english-gutenberg-poetry`

7. Conclusion

In the scope of this thesis, we research aesthetic emotions classification in poetry extensively. Our classification task uses PO-EMO by Haider et al. (2020) as the main resource. PO-EMO is a small dataset, consists of German and English poems. The poems are annotated with 9 emotion labels: *Suspense, Awe/Sublime, Sadness, Annoyance, Uneasiness, Beauty/Joy, Vitality, Humor* and *Nostalgia*. We model our classification task as a multi-label one. Due to the small amount of samples available, we do not classify on *Nostalgia*.

We research three approaches to build our classifier in Chapter 4 and decide on using BERT by Devlin et al. (2019), a state-of-the-art deep learning model as the baseline. After that, we perform various experiments to improve on this baseline, such as transfer learning (Section 5.1), data augmentation (Section 5.2) and cross-lingual learning (Section 5.3).

Our transfer learning experiments are based largely on *fine-tuning* BERT, which is the process of introducing the pre-trained BERT model with additional data to increase performance on our classification. Our experiments show that data that comes from the same domain has the most positive impact on this, as experiments with poetry data outperform ones with data from another literature source by a good amount. We also show that *intermediate task training*, introduced by Phang, Févry, and Bowman (2019), is a helpful extension to the BERT pipeline. We share the same observation as Pruksachatkun et al. (2020), however, with regard to how intermediate task training still need to be researched in more detail for a better explanation of its mechanism and benefits.

Data augmentation is the process of artificially increase the amount of data in low-resource setting by using transformation technique. Due to the unique aesthetic and formal constraints of poetry, data augmentation becomes an interesting venue for our experiments to explore. We find out that creating samples that can maintain the poetic form has better results than breaking it as demonstrated by the back-translation (Section 5.2.2, Section 5.2.3 and Section 5.2.4). Adding duplicated sample seems to help our classifiers improve too, although not by much.

Cross-lingual experiments can be considered the least performing experiments in our thesis. We use the German data available in PO-EMO and combine them with techniques from our other experiments, but the results are not very convincing. We have to use m-BERT instead BERT-large in these experiments to accommodate cross-lingual ability, which can be a reason for the disappointing results. We acknowledge the failure in these experiments, and would like to revisit them in near future if given the time.

Our experiments work relatively well individually, but not very well when combined together. We therefore use ensemble of classifiers' output instead, which improve results overall. We use the best model combination from our research to perform poetry analysis in Section 6.2. Although unable to draw any conclusive observation, it opens up possibilities for further experiments in NLP with poetry as a research domain.

Appendices

A. Dataset Statistics

Label	Angry	Disgusted	Fearful	Happy	Neutral	Sad	+ Surprised	- Surprised
Line counts	872	450	842	1820	12039	1035	383	636

Table A.1.: Statistics for the tales-emotion corpus (Section 5.1.2). **+ Surprised**: positively surprised. **- Surprised**: negatively surprised.

B. Ensemble of Pairs of Models

poetry pretraining	0.50									
meter finetuning	0.44	0.48								
emotion finetuning	0.39	0.40	0.43							
stanza shuffling	0.42	0.44	0.40	0.46						
over sampling	0.36	0.38	0.32	0.37	0.38					
m-BERT de	0.38	0.36	0.32	0.34	0.28	0.41				
m-BERT finetune	0.39	0.39	0.35	0.36	0.32	0.32	0.42			
m-BERT finetune_cl	0.37	0.36	0.32	0.32	0.30	0.35	0.31	0.43		
word replacing	0.43	0.42	0.37	0.40	0.37	0.34	0.37	0.32	0.48	
back translation	0.44	0.42	0.38	0.41	0.35	0.36	0.37	0.35	0.39	0.46
	poetry pretraining	meter finetuning	emotion finetuning	stanza shuffling	over sampling	m-BERT_de	m-BERT finetune	m-BERT finetune_cl	word replacing	back translation

Table B.1.: Average macro F1-score across 2 splits with INTERSECTION operation. Average F1-score across 2 splits for the baseline BERT models is 0.37.

poetry pretraining	0.50									
meter finetuning	0.52	0.48								
emotion finetuning	0.51	0.49	0.43							
stanza shuffling	0.54	0.51	0.48	0.47						
over sampling	0.54	0.50	0.48	0.49	0.38					
m-BERT de	0.51	0.50	0.48	0.51	0.49	0.41				
m-BERT finetune	0.52	0.50	0.48	0.51	0.49	0.49	0.42			
m-BERT finetune_cl	0.53	0.52	0.50	0.53	0.51	0.47	0.51	0.43		
word replacing	0.54	0.52	0.52	0.53	0.51	0.52	0.52	0.55	0.48	
back translation	0.52	0.50	0.49	0.51	0.51	0.50	0.50	0.51	0.54	0.46
	poetry pretraining	meter finetuning	emotion finetuning	stanza shuffling	over sampling	m-BERT_de	m-BERT finetune	m-BERT finetune_cl	word replacing	back translation

Table B.2.: Average macro F1-score across 2 splits with UNION operation. Average F1-score across 2 splits for the baseline BERT models is 0.37.

poetry pretraining	0.50									
meter finetuning	0.49	0.48								
emotion finetuning	0.43	0.46	0.43							
stanza shuffling	0.50	0.48	0.44	0.47						
over sampling	0.44	0.44	0.41	0.44	0.38					
m-BERT de	0.41	0.42	0.37	0.40	0.37	0.41				
m-BERT finetune	0.46	0.50	0.42	0.46	0.37	0.38	0.42			
m-BERT finetune_cl	0.45	0.50	0.38	0.43	0.38	0.42	0.43	0.43		
word replacing	0.50	0.49	0.44	0.47	0.48	0.43	0.43	0.43	0.48	
back translation	0.48	0.49	0.45	0.48	0.44	0.42	0.47	0.44	0.47	0.46
	poetry pretraining	meter finetuning	emotion finetuning	stanza shuffling	over sampling	m-BERT_de	m-BERT finetune	m-BERT finetune_cl	word replacing	back translation

Table B.3.: Average macro F1-score across 2 splits with AVERAGE operation. Average F1-score across 2 splits for the baseline BERT models is 0.37.

C. Ensemble of Triples of Models

Due to the large number of model combinations possible, we only report the top 10 best performing models in this Section.

Average F1-macro	Model 1	Model 2	Model 3
0.40	back-translation	meter_finetuning	poetry_pretraining
0.40	meter_finetuning	poetry_pretraining	stanza_shuffling
0.39	back-translation	meter_finetuning	stanza_shuffling
0.38	back-translation	poetry_pretraining	stanza_shuffling
0.38	meter_finetuning	poetry_pretraining	word_replacing
0.38	meter_finetuning	stanza_shuffling	word_replacing
0.37	back-translation	poetry_pretraining	word_replacing
0.37	emotion_finetuning	meter_finetuning	poetry_pretraining
0.37	emotion_finetuning	meter_finetuning	stanza_shuffling
0.37	poetry_pretraining	stanza_shuffling	word_replacing

Table C.1.: Top 10 average macro F1-score for INTERSECTION operation and corresponding model triples. Average F1-score across 2 splits for the baseline BERT models is 0.37.

Average F1-macro	Model 1	Model 2	Model 3
0.56	mbert_finetune_classification	stanza_shuffling	word_replacing
0.55	mbert_de_finetune	mbert_finetune_classification	word_replacing
0.55	mbert_de_finetune	oversampling	poetry_pretraining
0.55	mbert_de_finetune	stanza_shuffling	word_replacing
0.55	mbert_finetune_classification	oversampling	word_replacing
0.55	oversampling	poetry_pretraining	stanza_shuffling
0.55	oversampling	poetry_pretraining	word_replacing
0.55	poetry_pretraining	stanza_shuffling	word_replacing
0.54	back-translation	mbert_finetune_classification	word_replacing
0.54	back-translation	oversampling	word_replacing

Table C.2.: Top 10 average macro F1-score for UNION operation and corresponding model triples. Average F1-score across 2 splits for the baseline BERT models is 0.37.

Average F1-macro	Model 1	Model 2	Model 3
0.51	back-translation	meter_finetuning	word_replacing
0.51	meter_finetuning	poetry_pretraining	word_replacing
0.5	back-translation	mbert_de_finetune	poetry_pretraining
0.5	back-translation	meter_finetuning	oversampling
0.5	back-translation	poetry_pretraining	stanza_shuffling
0.5	back-translation	poetry_pretraining	word_replacing
0.5	meter_finetuning	stanza_shuffling	word_replacing
0.49	back-translation	mbert_de	poetry_pretraining
0.49	back-translation	mbert_finetune_classification	stanza_shuffling
0.49	back-translation	meter_finetuning	poetry_pretraining

Table C.3.: Top 10 average macro F1-score for AVERAGE operation and corresponding model triples. Average F1-score across 2 splits for the baseline BERT models is 0.37.

Bibliography

[1] Cecilia Alm, Dan Roth, and Richard Sproat. "Emotions from Text: Machine Learning for Text-based Emotion Prediction." In: Jan. 2005. DOI: 10.3115/1220575.1220648.

[2] Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. "Emotions from Text: Machine Learning for Text-based Emotion Prediction". In: *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Vancouver, British Columbia, Canada: Association for Computational Linguistics, Oct. 2005, pp. 579–586. URL: https://www.aclweb.org/anthology/H05-1073.

[3] Ebba Cecilia Ovesdotter Alm. *Affect in* text and speech*. University of Illinois at Urbana-Champaign, 2008.

[4] Ouais Alsharif, Deema Alshamaa, and Nada Ghneim. "Emotion classification in Arabic poetry using machine learning". In: *International Journal of Computer Applications* 65.16 (2013).

[5] Saima Aman and Stan Szpakowicz. "Identifying Expressions of Emotion in Text". In: *TSD*. 2007.

[6] Mikel Artetxe and Holger Schwenk. *Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond*. 2019. arXiv: 1812.10464 [cs.CL].

[7] Iz Beltagy, Arman Cohan, and Kyle Lo. "SciBERT: Pretrained Contextualized Embeddings for Scientific Text". In: *CoRR* abs/1903.10676 (2019). arXiv: 1903.10676. URL: http://arxiv.org/abs/1903.10676.

[8] Laura-Ana-Maria Bostan and Roman Klinger. "An Analysis of Annotated Corpora for Emotion Classification in Text". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 2104–2119. URL: https://www.aclweb.org/anthology/C18-1179.

[9] Sven Buechel and Udo Hahn. "EmoBank: Studying the Impact of Annotation Perspective and Representation Format on Dimensional Emotion Analysis". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 578–585. URL: https://www.aclweb.org/anthology/E17-2092.

[10] N. V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357. ISSN: 1076-9757. DOI: 10.1613/jair.953. URL: http://dx.doi.org/10.1613/jair.953.

[11] Kevin Clark et al. *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators*. 2020. arXiv: 2003.10555 [cs.CL].

[12] Alexis Conneau et al. *Unsupervised Cross-lingual Representation Learning at Scale*. 2020. arXiv: 1911.02116 [cs.CL].

[13] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *ArXiv* abs/1810.04805 (2019).

[14] Thomas G Dietterich. “Ensemble methods in machine learning”. In: *International workshop on multiple classifier systems*. Springer. 2000, pp. 1–15.

[15] Paul Ekman. “An argument for basic emotions”. In: *Cognition and Emotion* 6.3-4 (1992), pp. 169–200. DOI: 10.1080/02699939208411068. eprint: https://doi.org/10.1080/02699939208411068. URL: https://doi.org/10.1080/02699939208411068.

[16] Alex Estes and Christopher Hench. “Supervised Machine Learning for Hybrid Meter”. In: *Proceedings of the Fifth Workshop on Computational Linguistics for Literature*. San Diego, California, USA: Association for Computational Linguistics, June 2016, pp. 1–8. DOI: 10.18653/v1/W16-0201. URL: https://www.aclweb.org/anthology/W16-0201.

[17] Diman Ghazi, Diana Inkpen, and Stan Szpakowicz. “Detecting Emotion Stimuli in Emotion-Bearing Sentences”. In: *CICLing*. 2015.

[18] Thomas Haider et al. “PO-EMO: Conceptualization, Annotation, and Modeling of Aesthetic Emotions in German and English Poetry”. In: *LREC*. 2020.

[19] Zhiting Hu et al. “Harnessing Deep Neural Networks with Logic Rules”. In: *ArXiv* abs/1603.06318 (2016).

[20] Harsh Jhamtani et al. “Learning Rhyming Constraints using Structured Adversaries”. In: *EMNLP/IJCNLP*. 2019.

[21] Andrew Johnson et al. “Cross-lingual Transfer Learning for Japanese Named Entity Recognition”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 182–189. DOI: 10.18653/v1/N19-2023. URL: https://www.aclweb.org/anthology/N19-2023.

[22] Giannis Karamanolakis, Daniel Hsu, and Luis Gravano. *Cross-Lingual Text Classification with Minimal Resources by Transferring a Sparse Teacher*. 2020. arXiv: 2010.02562 [cs.CL].

[23] Jasleen Kaur and Jatinderkumar R Saini. “Emotion detection and sentiment analysis in text corpus: a differential study with informal and formal writing styles”. In: *International Journal of Computer Application, ISSN* (2014), pp. 0975–8887.

[24] Yunsu Kim, Yingbo Gao, and Hermann Ney. “Effective Cross-lingual Transfer of Neural Machine Translation Models without Shared Vocabularies”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 1246–1257. DOI: 10.18653/v1/P19-1120. URL: https://www.aclweb.org/anthology/P19-1120.

[25] Sosuke Kobayashi. “Contextual Augmentation: Data Augmentation by Words with Paradigmatic Relations”. In: *ArXiv* abs/1805.06201 (2018).

[26] Maximilian Köper, Evgeny Kim, and Roman Klinger. “IMS at EmoInt-2017: Emotion Intensity Prediction with Affective Norms, Automatically Extended Resources and Deep Learning”. In: *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 50–57. DOI: 10.18653/v1/W17-5206. URL: https://www.aclweb.org/anthology/W17-5206.

[27] Florian Kunneman, Christine Liebrecht, and Antal van den Bosch. “The (Un)Predictability of Emotional Hashtags in Twitter”. In: *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)*. Gothenburg, Sweden: Association for Computational Linguistics, Apr. 2014, pp. 26–34. DOI: 10.3115/v1/W14-1304. URL: https://www.aclweb.org/anthology/W14-1304.

[28] Guillaume Lample and Alexis Conneau. *Cross-lingual Language Model Pretraining*. 2019. arXiv: 1901.07291 [cs.CL].

[29] Jey Han Lau et al. "Deep-speare: A joint neural model of poetic language, meter and rhyme". In: *ArXiv* abs/1807.03491 (2018).

[30] Jinhyuk Lee et al. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". In: *CoRR* abs/1901.08746 (2019). arXiv: 1901.08746. URL: http://arxiv.org/abs/1901.08746.

[31] Panpan Li et al. "Short Text Emotion Analysis Based on Recurrent Neural Network". In: *Proceedings of the 6th International Conference on Information Engineering*. ICIE '17. Dalian Liaoning, China: Association for Computing Machinery, 2017. ISBN: 9781450352109. DOI: 10.1145/3078564.3078569. URL: https://doi.org/10.1145/3078564.3078569.

[32] Yanran Li et al. "DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset". In: *ArXiv* abs/1710.03957 (2017).

[33] Jasy Suet Yan Liew and Howard R. Turtle. "Exploring Fine-Grained Emotion Detection in Tweets". In: *Proceedings of the NAACL Student Research Workshop*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 73–80. DOI: 10.18653/v1/N16-2011. URL: https://www.aclweb.org/anthology/N16-2011.

[34] Dayiheng Liu et al. "A Multi-Modal Chinese Poetry Generation Model". In: *2018 International Joint Conference on Neural Networks (IJCNN)* (2018), pp. 1–8.

[35] V. Liu, C. Banea, and R. Mihalcea. "Grounded emotions". In: *2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*. 2017, pp. 477–483.

[36] Xuezhe Ma and Eduard H. Hovy. "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF". In: *ArXiv* abs/1603.01354 (2016).

[37] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[38] Saif M. Mohammad. "Obtaining Reliable Human Ratings of Valence, Arousal, and Dominance for 20, 000 English Words". In: *ACL*. 2018.

[39] Jose M Moyano et al. "Review of ensembles of multi-label classifiers: models, experimental study and prospects". In: *Information Fusion* 44 (2018), pp. 33–45.

[40] Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. *BERTweet: A pre-trained language model for English Tweets*. 2020. arXiv: 2005.10200 [cs.CL].

[41] Vu-Linh Nguyen et al. *On Aggregation in Ensembles of Multilabel Classifiers*. 2020. arXiv: 2006.11916 [cs.LG].

[42] Christian Obermeier et al. "Aesthetic and emotional effects of meter and rhyme in poetry". In: *Frontiers in psychology* 4 (2013), p. 10.

[43] S. J. Pan and Q. Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359.

[44] Matthew E. Peters et al. *Deep contextualized word representations*. 2018. arXiv: 1802.05365 [cs.CL].

[45] Jason Phang, Thibault Févry, and Samuel R. Bowman. *Sentence Encoders on STILTs: Supplementary Training on Intermediate Labeled-data Tasks*. 2019. arXiv: 1811.01088 [cs.CL].

[46] Telmo Pires, Eva Schlinger, and Dan Garrette. “How Multilingual is Multilingual BERT?” In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 4996–5001. DOI: 10.18653/v1/P19-1493. URL: https://www.aclweb.org/anthology/P19-1493.

[47] Daniel Preoţiuc-Pietro et al. “Modelling Valence and Arousal in Facebook posts”. In: *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 9–15. DOI: 10.18653/v1/W16-0404. URL: https://www.aclweb.org/anthology/W16-0404.

[48] Yada Pruksachatkun et al. *Intermediate-Task Transfer Learning with Pretrained Models for Natural Language Understanding: When and Why Does It Work?* 2020. arXiv: 2005.00628 [cs.CL].

[49] Jesse Read et al. “Classifier chains for multi-label classification”. In: *Machine learning* 85.3 (2011), p. 333.

[50] Sravana Reddy and Kevin Knight. “Unsupervised discovery of rhyme schemes”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 2011, pp. 77–82.

[51] Nils Reimers and Iryna Gurevych. “Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging”. In: *ArXiv* abs/1707.09861 (2017).

[52] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “”Why Should I Trust You?”: Explaining the Predictions of Any Classifier”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 2016, pp. 1135–1144.

[53] Lior Rokach, Alon Schclar, and Ehud Itach. “Ensemble methods for multi-label classification”. In: *Expert Systems with Applications* 41.16 (2014), pp. 7507–7523.

[54] Klaus R. Scherer and Harald Günter Wallbott. “Evidence for universality and cultural variation of differential emotion response patterning.” In: *Journal of personality and social psychology* 66 2 (1994), pp. 310–28.

[55] Ines Schindler et al. “Measuring aesthetic emotions: A review of the literature and a new assessment tool”. In: *PLOS ONE* 12.6 (June 2017), pp. 1–45. DOI: 10.1371/journal.pone.0178899. URL: https://doi.org/10.1371/journal.pone.0178899.

[56] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Improving Neural Machine Translation Models with Monolingual Data”. In: *ArXiv* abs/1511.06709 (2016).

[57] Boaz Shmueli and Lun-Wei Ku. “SocialNLP EmotionX 2019 Challenge Overview: Predicting Emotions in Spoken Dialogues and Chats”. In: *ArXiv* abs/1909.07734 (2019).

[58] Morgan Sonderegger. “Applications of Graph Theory to an English Rhyming Corpus”. In: *Comput. Speech Lang.* 25.3 (July 2011), pp. 655–678. ISSN: 0885-2308. DOI: 10.1016/j.csl.2010.05.005. URL: https://doi.org/10.1016/j.csl.2010.05.005.

[59] Chuanqi Tan et al. “A Survey on Deep Transfer Learning”. In: *ICANN*. 2018.

[60] Grigorios Tsoumakas and Ioannis Vlahavas. “Random k-labelsets: An ensemble method for multilabel classification”. In: *European conference on machine learning*. Springer. 2007, pp. 406–417.

[61] Grigorios Tsoumakas et al. “Correlation-based pruning of stacked binary relevance models for multi-label learning”. In: *Proceedings of the 1st international workshop on learning from multi-label data*. 2009, pp. 101–116.

[62] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[63] William Yang Wang and Diyi Yang. "That's So Annoying!!!: A Lexical and Frame-Semantic Embedding Based Data Augmentation Approach to Automatic Categorization of Annoying Behaviors using #petpeeve Tweets". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 2557–2563. DOI: 10.18653/v1/D15-1306. URL: https://www.aclweb.org/anthology/D15-1306.

[64] Jason Wei and Kai Zou. "EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6382–6388. DOI: 10.18653/v1/D19-1670. URL: https://www.aclweb.org/anthology/D19-1670.

[65] Shijie Wu and Mark Dredze. "Beto, Bentz, Becas: The Surprising Cross-Lingual Effectiveness of BERT". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 833–844. DOI: 10.18653/v1/D19-1077. URL: https://www.aclweb.org/anthology/D19-1077.

[66] Y. Wu et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *ArXiv* abs/1609.08144 (2016).

[67] Jiateng Xie et al. "Neural Cross-Lingual Named Entity Recognition with Minimal Resources". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 369–379. DOI: 10.18653/v1/D18-1034. URL: https://www.aclweb.org/anthology/D18-1034.

[68] Qizhe Xie et al. "Unsupervised Data Augmentation for Consistency Training". In: *arXiv: Learning* (2019).

[69] Adams Wei Yu et al. "QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension". In: *ArXiv* abs/1804.09541 (2018).

[70] Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-Level Convolutional Networks for Text Classification". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 649–657.

[71] Barret Zoph et al. "Transfer Learning for Low-Resource Neural Machine Translation". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1568–1575. DOI: 10.18653/v1/D16-1163. URL: https://www.aclweb.org/anthology/D16-1163.