Teaching Machines about Emotions

by

Bjarke Felbo

Bachelor of Science in Engineering (Mathematics and Technology),
Technical University of Denmark (2016)

Submitted to the Program in Media Arts and Sciences
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

Signature redacted

Author ....
Program in Media Arts and Sciences
May 11th, 2018

Signature redacted

Certified by..............................
Iyad Rahwan
Associate Professor
Program in Media Arts and Sciences
Thesis Supervisor

Signature redacted

Accepted by.....................
Tod Machover
Academic Head
Program in Media Arts and Sciences

Teaching Machines about Emotions

by

Bjarke Felbo

Submitted to the Program in Media Arts and Sciences
on May 11th, 2018, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

Artificial intelligence algorithms are becoming an increasingly important part of human life with many chat bots and digital personal assistants now interacting directly with us through natural language. Such human-computer interaction can be made more useful by enriching the underlying algorithms with a detailed sense of emotion. In my thesis I propose new ways to detect, encode and modify emotional content in text. First, I show how we can leverage the vast amount of texts on social media with emojis to train a classifier that can accurately detect various kinds of emotional content in text. Secondly, I introduce a state-of-the-art domain adaptation method that is explicitly designed to tackle issues occurring in the messy real-world text data that existing NLP methods struggle with. Lastly, I propose a new algorithm that could be used to decompose text inputs into disentangled representations and then manipulate these representations in a controlled manner to obtain a modified version of the input.

Thesis Supervisor: Iyad Rahwan
Title: Associate Professor
Program in Media Arts and Sciences

Teaching Machines about Emotions

by

Bjarke Felbo

The following people served as readers for this thesis:

Signature redacted

Thesis Reader ....

[illegible]

Hugo Larochelle
Research Scientist
Google Brain

Signature redacted

Thesis Reader ....

Alex 'Sandy' Pentland
Professor
MIT Media Lab

Preface

Throughout the last two years, I've been captivated by the idea of using natural language processing (NLP) algorithms to study large-scale social phenomena related to language. As I naïvely tried to study racism on social media, I quickly learned the limitations of existing methods and how difficult it can be to do proper studies of human behavior using NLP algorithms. Motivated by these shortcomings I ended up – as many researchers before me – focusing on the tools rather than the topic itself. Nevertheless, I found that I truly enjoyed designing these tools and that it was a great motivation to build tools that could potentially empower social scientists to carry out many important projects in the future. My research in this area is compressed into the following three papers, which this thesis heavily borrows from:

- **Bjarke Felbo**, Alan Mislove, Anders Søgaard, Iyad Rahwan and Sune Lehmann. *Using millions of emoji occurrences to learn any-domain representations for detecting emotion, sentiment and sarcasm.* Empirical Methods in Natural Language Processing (EMNLP) 2017 long paper (oral).

- **Bjarke Felbo***, Michiel Bakker*, Abhimanyu Dubey, Sadhika Malladi, Alex 'Sandy' Pentland, Iyad Rahwan. *Towards Real-World Domain Adaptation for Text through Prediction Propagation.* Under review.

- **Bjarke Felbo**, Remi Mir, Iyad Rahwan. *Disentangling Content and Style through Representation Distillation.* Work in progress.

* shared first authorship

Acknowledgements

My two years at MIT have been an absolutely amazing experience! Being a Dane from the cold north, this is not something I say lightly. I attribute this wonderful experience entirely to the great people I've been lucky enough to have around me.

I'd like to thank my amazing advisor, Iyad Rahwan, for giving me the freedom to explore while guiding me towards the grand questions that are too often left unexplored. It's also been fantastic to work with Nick Obradovich, who's been ready to challenge my crazy ideas with a smile and teach me how to do the best possible science. Sune Lehmann has always been there to help me navigate academia, which I'm very thankful for. I'd also like to thank Yves-Alexandre de Montjoye for teaching me how to tackle research problems back when I had no clue at all. A big thanks goes out to all the collaborators I've had at MIT and elsewhere – you've continued to impress me with your commitment and insights.

I've been so happy to be in the Scalable Cooperation research group. Not only have the group challenged me intellectually, but they've always been there for me. Thanks for all the ping pong matches, late night talks, flying stuffed animals, and other sillyness. It was perfect.

There's so many great people in the Media Lab that I won't start naming them all. I'm truly thankful for all the friendships I've made. Even when everyone were stressed out, you managed to find time for me and each other, which is so impressive.

I'd also like to thank Olav, Anne, Kjartan, Solveig, Mina, Ida, and all the others who have been there to cheer me on from afar when I was feeling down or when I just needed a new perspective on things.

Perhaps my most important thanks goes out to my roommates Andres, Alejandro and Florian. I already have so many fond memories with you guys that I couldn't imagine being without. You have truly earned your place as a part of my family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural language processing (NLP) algorithms are becoming an increasingly important part of human life. These algorithms empower us to – among other things – search vast amounts of information, communicate across language barriers, and identify interesting patterns in the billions of texts available online. With the emergence of consumer-facing artificial intelligence (AI) agents such as Apple's Siri that interact with us directly, NLP algorithms become even more central to our lives.

These human-computer interactions facilitated by NLP can be made more useful and engaging if the underlying algorithms have a detailed sense of emotion. Equally important, better algorithms for analyzing and manipulating emotional content in texts can allow us to better understand crucial social phenomena such as racism and bullying.

In this thesis, I explore different ways to design algorithms for modeling the rich spectrum of emotions that humans employ. A key focus for me when designing these new algorithms has been to make the algorithms learn about human language without explicitly labeled data. This is critical not only because obtaining annotated data is hard, but also because small datasets of labeled data can easily contain undesirable biases that make their way into the model. The algorithms I've designed can be used not only for emotion-related tasks, but also a variety of other tasks. However, for conciseness and focus, this thesis will only examine how the algorithms can applied for emotion-related tasks.

The organization of this thesis is oriented around three of my research projects. Chapter 2 first briefly introduces some useful background knowledge for understanding the research projects. Chapter 3 describes the way we used a billion emojis to learn state-of-the-art representations for detecting sentiment, emotions, and sarcasm. Chapter 4 demonstrates how our new Prediction Propagation method achieves state-of-the-art performance on challenging domain adaptation datasets. Chapter 5 proposes a new method that could potentially be used for disentangling content and style. Chapter 6 discusses the usefulness of the methods proposed in this thesis. Finally, Chapter 7 concludes on the projects and describes potential next steps.

Chapter 2

Background

2.1 Analyzing emotions expressed in language

The thesis describes methods that – like many others – attempt to model human emotions in text. However, before delving into the more technical parts of how to do this modeling, it's interesting to briefly consider what it is we mean by 'emotions'. Despite the apparent prevalence of emotions, there's no clear consensus in the academic community on fundamental questions such as what constitutes an emotion, the number of emotions, and the distinctions between them. The disagreements have had researchers form three dominant theories of emotion: Basic emotions theory, appraisal theory, and conceptual act theory [75]. The basic emotions theory argues that evolution has caused us to have six to eight fundamental emotions that we to some degree share with animals [92]. Appraisal theory argues that we feel emotions as we subconsciously evaluate a list of mental check points of how an event will affect us directly or indirectly [60]. Lastly, conceptual act theory argues that emotions are defined purely from the society and culture that each individual is exposed to and that emotions thus are learned constructs [5]. These one-line descriptions do not give these theories justice so I highly encourage the reader to read the provided references. Due to the ambiguity of what an emotion is, most machine learning papers rely on datasets labeled by human annotators without explicit instructions on how to distinguish between emotions. This thesis follows the same approach despite its potential issues.

Measuring emotions in text has long been of interest to the natural language processing (NLP) community. Measuring sentiment of texts on online forums as a way to understand consumers' opinions initially fueled corporate interest in this method [66]. Over time the techniques to measure sentiment evolved from dictionary approaches [10, 62] to machine learning methods like deep neural networks [104]. Many community-led competitions (e.g. [72]) have also helped establish sentiment analysis as a central task in NLP. Researchers have since modeled interesting related tasks such as emotion detection [81, 12], sarcasm detection [96, 43], and bullying [23]. Lack of clear definitions on what constitutes an emotion, a sarcastic comment, or bulling has made it difficult to obtain high-quality data, thereby limiting the ability to properly model these phenomena in practice and conduct studies using such models. These limitations are in part a motivation for why I in this thesis focus on designing algorithms without the need for explicitly annotated datasets.

In addition to the research on understanding emotional content in language, there's also been substantial research on other interesting ways of teaching machines about emotions. For instance, machine learning has been used to detect affective states [14, 83], to give AI agents a sense of emotions [1, 29], and to improve human-computer interaction [67, 70]. For brevity and focus, I refrain from going into details on these interesting research directions and focus exclusively on analyzing emotions using natural language processing (NLP) methods.

2.2 Neural networks and deep learning

The methods and algorithms proposed in this thesis builds on the deep learning framework, which has been massively successful for computer vision [50, 99] and natural language processing [85, 58]. The core idea is to use neural networks to learn distributed representations of the input data that are useful for a subsequent learning task [35, 63]. The neural network is composed of many non-linear layers, giving it a high degree of flexibility in shaping the representation as needed to perform the downstream task.

The easiest way for a neural network to learn a useful representation for a specific task is arguably to have a large labeled dataset, but there are also many other useful approaches. Autoencoders attempt to compress their input as a way to learn representations that capture the high-level dynamics [95, 15, 49]. Rather than reconstructing every part of the input, it can be better to teach the network through specific high-level reconstructions tasks, in this way forcing it to learn high-level representations [64, 102]. This thesis also explores how carefully designed reconstruction losses can be used as a way to learn useful representations without having labeled data.

A useful property of neural networks is that the representations computed at any given layer can be easily reused for other modeling purposes. One way to do this is to use a pretrained neural network as a feature extractor and then train a new machine learning model on top [24]. Alternatively, the neural network can have multiple output layers, each with their own loss for the specific label being predicted [18]. The flexibility of neural networks is key to their success and something that's being exploited in this thesis.

These deep learning models have millions of parameters, which are trained using backpropagation and stochastic gradient descent (SGD) [73]. Algorithms with adaptive learning [101, 48] are often used to reduce the optimization challenges, but nevertheless, some methods like the generative adversarial network (GAN) [34] are notoriously difficult to train due to the adversarial loss being used. This adversarial loss is also needed for the RepDistill method proposed in Chapter 5.

Chapter 3

Learning Representations from a Billion Emojis

A variety of NLP tasks are limited by scarcity of manually annotated data. Therefore, co-occurring emotional expressions have been used for distant supervision in social media sentiment analysis and related tasks to make the models learn useful text representations before modeling these tasks directly. For instance, the state-of-the-art approaches within sentiment analysis of social media data use positive/negative emoticons for training their models [22, 87]. Similarly, hashtags such as #anger, #joy, #happytweet, #ugh, #yuck and #fml have in previous research been mapped into emotional categories for emotion analysis [59].

Distant supervision on noisy labels often enables a model to obtain better performance on the target task. We show that extending the distant supervision to a more diverse set of noisy labels enables the models to learn richer representations of emotional content in text, thereby obtaining better performance on benchmarks for detecting sentiment, emotions and sarcasm. We show that the learned representation of a single pretrained model generalizes across 5 domains.

Emojis are not always a direct labeling of emotional content. For instance, a positive emoji may serve to disambiguate an ambiguous sentence or to complement an otherwise relatively negative text. Kunneman et al. [52] discuss a similar duality

Table 3.1: Example sentences scored by our model. For each text the top five most likely emojis are shown with the model's probability estimates.

I love mom's cooking	😋 49.1%	😍 8.8%	❤ 3.1%	☺ 3.0%	♡ 2.9%
I love how you never reply back..	😒 14.0%	😑 8.3%	😠 6.3%	😐 5.4%	💔 5.1%
I love cruising with my homies	😎 34.0%	👌 6.6%	✌ 5.7%	😌 4.1%	💯 3.8%
I love messing with yo mind!!	😜 17.2%	👿 11.8%	😏 8.0%	😉 6.4%	🙊 5.3%
I love you and now you're just gone..	💔 39.1%	😔 11.0%	😞 7.3%	😪 5.3%	😢 4.5%
This is shit	😠 7.0%	😡 6.4%	😞 6.0%	😒 6.0%	😤 5.8%
This is the shit	🎧 10.9%	🎶 9.7%	👌 6.5%	😎 5.7%	😏 4.8%

in the use of emotional hashtags such as *#nice* and *#lame*. Nevertheless, our work shows that emojis can be used to classify the emotional content of texts accurately in many cases. For instance, our DeepMoji model captures varied usages of the word 'love' as well as slang such as 'this is the shit' being a positive statement (see Table 3.1). We provide an online demo at deepmoji.mit.edu to allow others to explore the predictions of our model.

3.1 Related work using noisy labels

Using emotional expressions as noisy labels in text to counter scarcity of labels is not a new idea [71, 33]. Originally, binarized emoticons were used as noisy labels, but later also hashtags and emojis have been used. To our knowledge, previous research has always manually specified which emotional category each emotional expression belong to. Prior work has used theories of emotion such as Ekman's six basic emotions and Plutchik's eight basic emotions [59, 86].

Such manual categorization requires an understanding of the emotional content of each expression, which is difficult and time-consuming for sophisticated combinations of emotional content. Moreover, any manual selection and categorization is prone to misinterpretations and may omit important details regarding usage. In contrast, our approach requires no prior knowledge of the corpus and can capture diverse usage of 64 types of emojis (see Table 3.1 for examples and Figure 3-3 for how the model implicitly groups emojis).

Another way of automatically interpreting the emotional content of an emoji is to learn emoji embeddings from the words describing the emoji-semantics in official emoji tables [25]. This approach, in our context, suffers from two severe limitations: a) It requires emojis at test time while there are many domains with limited or no usage of emojis. b) The tables do not capture the dynamics of emoji usage, i.e., drift in an emoji's intended meaning over time.

Knowledge can be transferred from the emoji dataset to the target task in many different ways. In particular, multitask learning with simultaneous training on multiple datasets has shown promising results [18]. However, multitask learning requires access to the emoji dataset whenever the classifier needs to be tuned for a new target task. Requiring access to the dataset is problematic in terms of violating data access regulations. There are also issues from a data storage perspective as the dataset used for this research contains hundreds of millions of tweets (see Table 3.2). Instead we use transfer learning [6] as described in §3.4, which does not require access to the original dataset, but only the pretrained classifier.

3.2 Pretraining using emoji prediction

In many cases, emojis serve as a proxy for the emotional contents of a text. Therefore, pretraining on the classification task of predicting which emoji were initially part of a text can improve performance on the target task (see §3.7 for an analysis of why our pretraining helps). Social media contains large amounts of short texts with emojis that can be utilized as noisy labels for pretraining. Here, we use data from Twitter from January 1st 2013 to June 1st 2017, but any dataset with emoji occurrences could be used.

Only English tweets without URL's are used for the pretraining dataset. Our hypothesis is that the content obtained from the URL is likely to be important for understanding the emotional content of the text in the tweet. Therefore, we expect emojis associated with these tweets to be noiser labels than for tweets without URLs, and the tweets with URLs are thus removed.

Proper tokenization is important for generalization. All tweets are tokenized on a word-by-word basis. Words with 2 or more repeated characters are shortened to the same token (e.g. 'loool' and 'looooool' are tokenized such that they are treated the same). Similarly, we use a special token for all URLs (only relevant for benchmark datasets), user mentions (e.g. '@acl2017' and '@emnlp2017' are thus treated the same) and numbers. To be included in the training set the tweet must contain at least 1 token that is not a punctuation symbol, emoji or special token[1].

Many tweets contain multiple repetitions of the same emoji or multiple different emojis. In the training data, we address this in the following way. For each unique emoji type, we save a separate tweet for the pretraining with that emoji type as the label. We only save a single tweet for the pretraining per unique emoji type regardless of the number of emojis associated with the tweet. This data preprocessing allows the pretraining task to capture that multiple types of emotional content are associated with the tweet while making our pretraining task a single-label classification instead of a more complicated multi-label classification.

To ensure that the pretraining encourages the models to learn a rich understanding of emotional content in text rather than only emotional content associated with the most used emojis, we create a balanced pretraining dataset. The pretraining data is split into a training, validation and test set, where the validation and test set is randomly sampled in such a way that each emoji is equally represented. The remaining data is upsampled to create a balanced training dataset.

[1]Details available at github.com/bfelbo/deepmoji

3.3 Neural architecture for any-domain representations

With the millions of emoji occurrences available, we can train very expressive classifiers with limited risk of overfitting. We use a variant of the Long Short-Term Memory (LSTM) model that has been successful at many NLP tasks [39, 85]. Our DeepMoji model uses an embedding layer of 256 dimensions to project each word into a vector space. A hyperbolic tangent activation function is used to force each embedding dimension to be within $[-1, 1]$. To capture the context of each word we use two bidirectional LSTM layers with 1024 hidden units in each (512 in each direction). Finally, an attention layer that take all of these layers as input using skip-connections is used (see Figure 3-1 for an illustration).

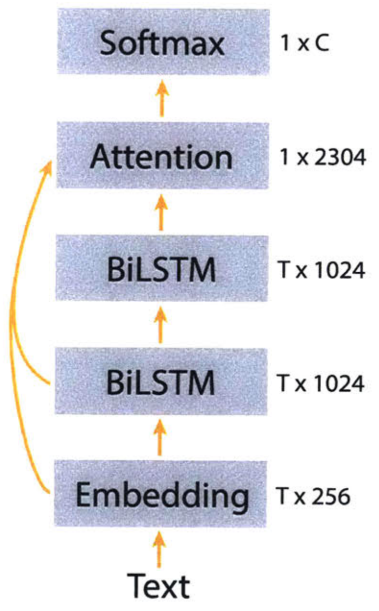

Figure 3-1: Illustration of the DeepMoji model with T being text length and C the number of classes.

The attention mechanism lets the model decide the importance of each word for the prediction task by weighing them when constructing the representation of the text. For instance, a word such as 'amazing' is likely to be very informative of the emotional meaning of a text and it should thus be treated accordingly. We use a simple approach inspired by [4, 100] with a single parameter pr. input channel:

$$
\begin{aligned}
e_t &= h_t w_a \\
a_t &= \frac{exp(e_t)}{\sum_{i=1}^{T} exp(e_i)} \\
v &= \sum_{i=1}^{T} a_i h_i
\end{aligned}
$$

Here h_t is the representation of the word at time step t and w_a is the weight matrix for the attention layer. The attention importance scores for each time step, a_t, are obtained by multiplying the representations with the weight matrix and then normalizing to construct a probability distribution over the words. Lastly, the representation vector for the text, v, is found by a weighted summation over all the time steps using the attention importance scores as weights. This representation vector obtained from the attention layer is a high-level encoding of the entire text, which is used as input to the final Softmax layer for classification. We find that adding the attention mechanism and skip-connections improves the model's capabilities for transfer learning (see §3.7 for more details).

The only regularization used for the pretraining task is a L2 regularization of 1E−6 on the embedding weights. For the finetuning additional regularization is applied (see §3.6). Our model is implemented using Theano [89] and we make an easy-to-use version available that uses Keras [17].

3.4 Sequential unfreezing for transfer learning

Our pretrained model can be fine-tuned to the target task in multiple ways with some approaches 'freezing' layers by disabling parameters updates to prevent overfitting. One common approach is to use the network as a feature extractor [24], where all layers in the model are frozen when fine-tuning on the target task except the last layer (hereafter referred to as the *'last'* approach). Alternatively, another common approach is to use the pretrained model as an initialization [26], where the full model is unfrozen (hereafter referred to as *'full'*).

We propose a new simple transfer learning approach, *'chain-thaw'*, that sequentially unfreezes and fine-tunes a single layer at a time. This approach increases accuracy on the target task at the expense of extra computational power needed for the fine-tuning. By training each layer separately the model is able to adjust the individual patterns across the network with a reduced risk of overfitting. The sequential fine-tuning seems to have a regularizing effect similar to what has been examined with layer-wise training in the context of unsupervised learning [26].

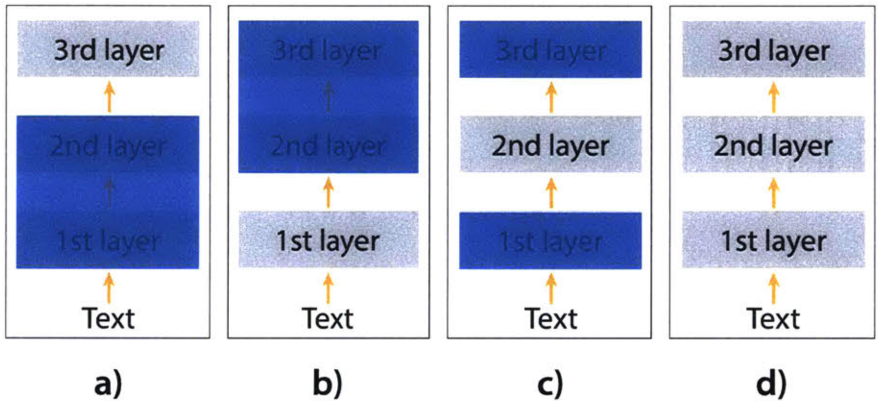

Figure 3-2: Illustration of the chain-thaw transfer learning approach, where each layer is fine-tuned separately. Layers covered with a blue rectangle are frozen. Step a) tunes any new layers, b) then tunes the 1st layer and c) the next layer until all layers have been fine-tuned individually. Lastly, in step d) all layers are fine-tuned together.

More specifically, the chain-thaw approach first fine-tunes any new layers (often only a Softmax layer) to the target task until convergence on a validation set. Then the approach fine-tunes each layer individually starting from the first layer in the network. Lastly, the entire model is trained with all layers. Each time the model converges as measured on the validation set, the weights are reloaded to the best setting, thereby preventing overfitting in a similar manner to early stopping [80]. This process is illustrated in Figure 3-2. Note how only performing step a) in the figure is identical to the 'last' approach, where the existing network is used as a feature extractor. Similarly, only doing step d) is identical to the 'full' approach, where the pretrained weights are used as an initialization for a fully trainable network. Although the chain-thaw procedure may seem extensive it is easily implemented with only a few lines of code. Similarly, the additional time spent on fine-tuning is limited when the target task uses GPUs on small datasets of manually annotated data as is often the case.

A benefit of the chain-thaw approach is the ability to expand the vocabulary to new domains with little risk of overfitting. For a given dataset up to 10000 new words from the training set are added to the vocabulary. §3.7 contains analysis on the added word coverage gained from this approach.

3.5 Importance of context for emoji prediction

We use a raw dataset of 56.6 billion tweets, which is then filtered to 1.2 billion relevant tweets (see details in §3.2). In the pretraining dataset a copy of a single tweet is stored once for each unique emoji, resulting in a dataset consisting of 1.6 billion tweets. Table 3.2 shows the distribution of tweets across different emoji types. To evaluate performance on the pretraining task a validation set and a test set both containing 640K tweets (10K of each emoji type) are used. The remaining tweets are used for the training set, which is balanced using upsampling.

Table 3.2: The number of tweets in the pretraining dataset associated with each emoji in millions.

😂	❤	😍	😭	😩	😊	😘	😒	👌	💕	♥	☺	😏	😁	😔	🙌
233.7	82.2	79.5	78.1	60.8	54.7	54.6	51.7	50.5	44.0	39.5	39.1	34.8	34.4	32.1	28.1
🙏	😳	💯	👍	✌	👏	🙈	😌	😴	😉	😎	💁	🎶	💖	💜	😅
24.8	23.4	21.6	21.0	20.5	20.3	19.9	19.6	18.9	17.5	17.0	16.9	16.1	15.3	15.2	15.0
👀	💙	😑	😋	😢	😫	🙊	😕	😄	😜	✋	😞	😡	😐	✨	💀
14.9	14.3	14.2	14.2	12.9	12.4	12.0	12.0	11.7	11.7	11.3	11.2	11.1	11.0	11.0	10.8
😈	👊	😪	💪	🔫	😤	💔	♡	😷	😣	😓	😖	🙅	😬	🎧	😠
10.2	9.6	9.5	9.3	9.2	8.9	8.7	8.6	8.1	6.3	6.0	5.7	5.6	5.5	5.4	5.1

Table 3.3: Accuracy of classifiers on the emoji prediction task. d refers to the dimensionality of each LSTM layer. Parameters are in millions.

	Params	Top 1	Top 5
Random	–	1.6%	7.8%
fasttext	12.8	12.8%	36.2%
DeepMoji (d = 512)	15.5	16.7%	43.3%
DeepMoji (d = 1024)	22.4	17.0%	43.8%

The performance of the DeepMoji model is evaluated on the pretraining task with the results shown in Table 3.3. Both top 1 and top 5 accuracy is used for the evaluation as the emoji labels are noisy with multiple emojis being potentially correct for any given sentence. For comparison we also train a version of our DeepMoji model with smaller LSTM layers and a bag-of-words classifier, fastText, that has recently shown competitive results [44]. We use 256 dimensions for this fastText classifier, thereby making it almost identical to only using the embedding layer from the DeepMoji model. The difference in top 5 accuracy between the fastText classifier (36.2%) and the largest DeepMoji model (43.8%) underlines the difficulty of the emoji prediction task. As the two classifiers only differ in that the DeepMoji model has LSTM layers and an attention layer between the embedding and Softmax layer, this difference in accuracy demonstrates the importance of capturing the context of each word.

3.6 Benchmarking across sentiment, emotion, and sarcasm tasks

We benchmark our method on 3 different NLP tasks using 8 datasets across 5 domains. To make for a fair comparison, we compare DeepMoji to other methods that also utilize external data sources in addition to the benchmark dataset. An averaged F1-measure across classes is used for evaluation in emotion analysis and sarcasm detection as these consist of unbalanced datasets while sentiment datasets are evaluated using accuracy.

An issue with many of the benchmark datasets is data scarcity, which is particularly problematic within emotion analysis. Recent papers proposing new methods for emotion analysis such as [81] only evaluate performance on a single benchmark dataset, SemEval 2007 Task 14, that contains 1250 observations. Recently, criticism has been raised concerning the use of correlation with continuous ratings as a measure [12], making only the somewhat limited binary evaluation possible. We only evaluate the emotions {Fear, Joy, Sadness} as the remaining emotions occur in less than 5% of the observations.

Table 3.4: Description of benchmark datasets. Datasets without pre-existing training/test splits are split by us (with splits publicly available). Data used for hyperparameter tuning is taken from the training set.

Identifier	Study	Task	Domain	Classes	N_{train}	N_{test}
SE0714	[82]	Emotion	Headlines	3	250	1000
Olympic	[79]	Emotion	Tweets	4	250	709
PsychExp	[97]	Emotion	Experiences	7	1000	6480
SS-Twitter	[91]	Sentiment	Tweets	2	1000	1113
SS-Youtube	[91]	Sentiment	Video Comments	2	1000	1142
SE1604	[61]	Sentiment	Tweets	3	7155	31986
SCv1	[96]	Sarcasm	Debate Forums	2	1000	995
SCv2-GEN	[65]	Sarcasm	Debate Forums	2	1000	2260

To fully evaluate our method on emotion analysis against the current methods we thus make use of two other datasets: A dataset of emotions in tweets related to the Olympic Games created by Sintsova et al. that we convert to a single-label classification task and a dataset of self-reported emotional experiences created by a large group of psychologists [97]. As these two datasets do not have prior evaluations, we evaluate against a state-of-the-art approach, which is based on a valence-arousal-dominance framework [12]. The scores extracted using this approach are mapped to the classes in the datasets using a logistic regression with parameter optimization using cross-validation. We release our preprocessing code and hope that these 2 two datasets will be used for future benchmarking within emotion analysis.

We evaluate sentiment analysis performance on three benchmark datasets. These small datasets are chosen to emphasize the importance of the transfer learning ability of the evaluated models. Two of the datasets are from SentiStrength [90], SS-Twitter and SS-Youtube, and follow the relabeling described in [74] to make the labels binary. The third dataset is from SemEval 2016 Task4A [61]. Due to tweets being deleted from Twitter, the SemEval dataset suffers from data decay, making it difficult to compare results across papers. At the time of writing, roughly 15% of the training dataset for SemEval 2016 Task 4A was impossible to obtain. We choose not to use review datasets for sentiment benchmarking as these datasets contain so many words pr. observation that even bag-of-words classifiers and unsupervised approaches can obtain a high accuracy [44, 69].

The current state of the art for sentiment analysis on social media (and winner of SemEval 2016 Task 4A) uses an ensemble of convolutional neural networks that are pretrained on a private dataset of tweets with emoticons, making it difficult to replicate [22]. Instead we pretrain a model with the hyperparameters of the largest model in their ensemble on the positive/negative emoticon dataset from Go et al. [33]. Using this pretraining as an initialization we finetune the model on the target tasks using early stopping on a validation set to determine the amount of training. We also implemented the Sentiment-Specific Word Embedding (SSWE) using the embeddings available on the authors' website [87], but found that it performed worse than the pretrained convolutional neural network. These results are therefore excluded.

For sarcasm detection we use the sarcasm dataset version 1 and 2 from the Internet Argument Corpus [96]. Note that results presented on these benchmarks in e.g. Oraby et al. [65] are not directly comparable as only a subset of the data is available online.[3] A state-of-the-art baseline is found by modeling the embedding-based features from Joshi et al. [43] alongside unigrams, bigrams and trigrams with an SVM. GoogleNews word2vec embeddings [58] are used for computing the embedding-based features. A hyperparameter search for regularization parameters is carried out using

[2]The authors report a higher accuracy in their paper, which is likely due to having a larger training dataset as they were able to obtain it before data decay occurred.

[3]We contacted the authors, but were unable to obtain the full dataset for neither version 1 or version 2.

Table 3.5: Comparison across benchmark datasets. Reported values are averages across five runs. Variations refer to transfer learning approaches in §3.4 with 'new' being a model trained without pretraining.

Dataset	Measure	Existing	DeepMoji (new)	DeepMoji (full)	DeepMoji (last)	DeepMoji (chainthaw)
SE0714	F1	.34 [Buechel]	.21	.31	.36	**.37**
Olympic	F1	.50 [Buechel]	.43	.50	**.61**	**.61**
PsychExp	F1	.45 [Buechel]	.32	.42	.56	**.57**
SS-Twitter	Acc	.82 [Deriu]	.62	.85	.87	**.88**
SS-Youtube	Acc	.86 [Deriu]	.75	.88	.92	**.93**
SE1604	Acc	.51 [Deriu]^2	.51	.54	**.58**	**.58**
SCv1	F1	.63 [Joshi]	.67	.65	.68	**.69**
SCv2-GEN	F1	.72 [Joshi]	.71	.71	.74	**.75**

cross-validation. Note that the sarcasm dataset version 2 contains both a quoted text and a sarcastic response, but to keep the models identical across the datasets only the response is used.

For training we use the Adam optimizer [48] with gradient clipping of the norm to 1. Learning rate is set to 1E−3 for training of all new layers and 1E−4 for finetuning any pretrained layers. To prevent overfitting on the small datasets, 10% of the channels across all words in the embedding layer are dropped out during training. Unlike e.g. [30] we do not drop out entire words in the input as some of our datasets contain observations with so few words that it could change the meaning of the text. In addition to the embedding dropout, L2 regularization for the embedding weights is used and 50% dropout is applied to the penultimate layer.

Table 3.5 shows that the DeepMoji model outperforms the state of the art across all benchmark datasets and that our new 'chain-thaw' approach consistently yields the highest performance for the transfer learning, albeit often only slightly better or equal to the 'last' approach. Results are averaged across 5 runs to reduce the variance. We test the statistical significance of our results by comparing the performance of DeepMoji (chain-thaw) vs. the state of the art. Bootstrap testing with 10000 samples is used. Our results are statistically significantly better than the state of the art with

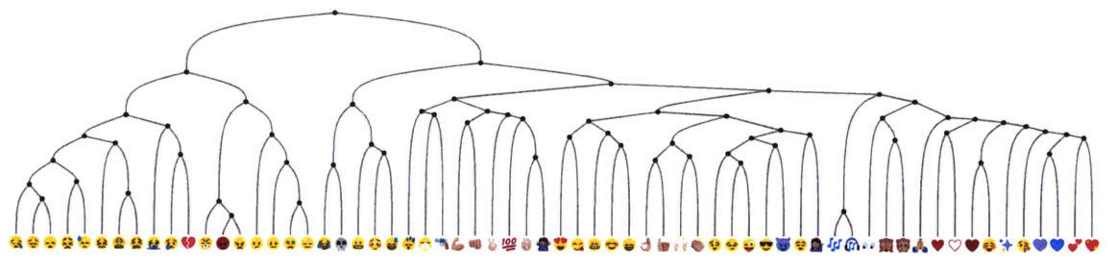

Figure 3-3: Hierarchical clustering of the DeepMoji model's predictions across categories on the test set. The dendrogram shows how the model learns to group emojis into overall categories and subcategories based on emotional content. The y-axis is the distance on the correlation matrix of the model's predictions measured using average linkage.

$p < 0.001$ on every benchmark dataset.

Our model is able to out-perform the state-of-the-art on datasets that originate from domains that differ substantially from the tweets on which it was pretrained. A key difference between the pretraining dataset and the benchmark datasets is the length of the observations. The average number of tokens pr. tweet in the pretraining dataset is 11, whereas e.g. the board posts from the Internet Argument Corpus version 1 [65] has an average of 66 tokens with some observations being much longer.

3.7 Analysis of DeepMoji modeling choices

One of the major differences between this work compared to previous papers using distant supervision is the diversity of the noisy labels used (see §3.1). For instance, both Deriu et al. [22] and Tang et al. [87] only used positive and negative emoticons as noisy labels. Other instances of previous work have used slightly more nuanced sets of noisy labels (see §3.1), but to our knowledge our set of noisy labels is the most diverse yet. To analyze the effect of using a diverse emoji set we create a subset of our pretraining data containing tweets with one of 8 emojis that are similar to the positive/negative emoticons used by Tang et al. [87] and Hu et al. [40]. As the dataset based on this reduced set of emojis contains 433M tweets, any difference in performance on benchmark datasets is likely linked to the diversity of labels rather than differences in dataset sizes.

We train our DeepMoji model to predict whether the tweets contain a positive or negative emoji and evaluate this pretrained model across the benchmark datasets. We refer to the model trained on the subset of emojis as *DeepMoji-PosNeg* (as opposed to *DeepMoji*). To test the emotional representations learned by the two pretrained models the 'last' transfer learning approach is used for the comparison, thereby only allowing the models to map already learned features to classes in the target dataset. Table 3.6 shows that DeepMoji-PosNeg yields lower performance compared to DeepMoji across all 8 benchmarks, thereby showing that the diversity of our emoji types encourage the model to learn a richer representation of emotional content in text that is more useful for transfer learning.

Table 3.6: Benchmarks using a smaller emoji set (Pos/Neg emojis) or a classic architecture (standard LSTM). Results for DeepMoji from Table 3.5 are added for convenience. Evaluation metrics are as in Table 3.5. Reported values are the averages across five runs.

Dataset	Pos/Neg emojis	Standard LSTM	DeepMoji
SE0714	.32	.35	.36
Olympic	.55	.57	.61
PsychExp	.40	.49	.56
SS-Twitter	.86	.86	.87
SS-Youtube	.90	.91	.92
SE1604	.56	.57	.58
SCv1	.66	.66	.68
SCv2-GEN	.72	.73	.74

Many of the emojis carry similar emotional content, but have subtle differences in usage that our model is able to capture. Through hierarchical clustering on the correlation matrix of the DeepMoji model's predictions on the test set we can see that the model captures many similarities that one would intuitively expect (see Figure 3-3). For instance, the model groups emojis into overall categories associated with e.g. negativity, positivity or love. Similarly, the model learns to differentiate within these categories, mapping sad emojis in one subcategory of negativity, annoyed in another subcategory and angry in a third one.

Our DeepMoji model architecture as described in §3.3 use an attention mechanism and skip-connections to ease the transfer of the learned representation to new domains and tasks. Here we compare the DeepMoji model architecture to that of a standard 2-layer LSTM, both compared using the 'last' transfer learning approach. We use the same regularization and training parameters.

As seen in Table 3.6 the DeepMoji model performs better than a standard 2-layer LSTM across all benchmark datasets. The two architectures performed equally on the pretraining task, suggesting that while the DeepMoji model architecture is indeed better for transfer learning, it may not necessarily be better for single supervised classification task with ample available data.

A reasonable conjecture is that the improved transfer learning performance is due to two factors: a) the attention mechanism with skip-connections provide easy access to learned low-level features for any time step, making it easy to use this information if needed for a new task b) the improved gradient-flow from the output layer to the early layers in the network due to skip-connections [36] is important when adjusting parameters in early layers as part of transfer learning to small datasets. Detailed analysis of whether these factors actually explain why our architecture outperform a standard 2-layer LSTM is left for future work.

Performance on the target task benefits strongly from pretraining as shown in Table 3.5 by comparing DeepMoji (new) to DeepMoji (chain-thaw). In this section we experimentally decompose the benefit of pretraining into 2 effects: word coverage and phrase coverage. These two effects help regularize the model by preventing overfitting.

There are numerous ways to express a specific sentiment, emotion or sarcastic comment. Consequently, the test set may contain specific language use not present in the training set. The pretraining helps the target task models attend to low-support evidence by having previously observed similar usage in the pretraining dataset. We first examine this effect by measuring the improvement in word coverage on the test set when using the pretraining with word coverage being defined as the % of words in the test dataset seen in the training/pretraining dataset (see Table 3.7). An important reason why the 'chain-thaw' approach outperforms other transfer learning approaches is that the embedding layer can be tuned with limited risk of overfitting. Table 3.7 shows the increased word coverage from adding new words to the vocabulary as part of that tuning (examine 'combined' vs. 'pretrained').

Note that word coverage can be a misleading metric in this context as for many of these small datasets a word will often occur only once in the training set. In contrast, all of the words in the pretraining vocabulary are present in thousands (if not millions) of observations in the emoji pretraining dataset thus making it possible for the model to learn a good representation of the emotional and semantic meaning. The added benefit of pretraining for learning word representations therefore likely extends beyond the differences seen in Table 3.7.

Table 3.7: Word coverage on benchmark test sets using only the vocabulary generated by finding words in the training data, the pretraining vocabulary or a combination of both vocabularies.

Dataset	Own	Pretrained	Combined
SE0714	41.9%	93.6%	94.0%
Olympic	73.9%	90.3%	96.0%
PsychExp	85.4%	98.5%	98.8%
SS-Twitter	80.1%	97.1%	97.2%
SS-Youtube	79.6%	97.2%	97.3%
SE1604	86.1%	96.6%	97.0%
SCv1	88.7%	97.3%	98.0%
SCv2-GEN	86.5%	97.2%	98.0%

To examine the importance of capturing phrases and the context of each word, we evaluate the accuracy on the SS-Youtube dataset using a fastText classifier pretrained on the same emoji dataset as our DeepMoji model. This fastText classifier is almost identical to only using the embedding layer from the DeepMoji model. We evaluate the representations learned by fine-tuning the models as feature extractors (i.e. using the 'last' transfer learning approach). The fastText model achieves an accuracy of 63% as compared to 93% for our DeepMoji model, thereby emphasizing the importance of phrase coverage. One concept that the LSTM layers likely learn is negation, which is known to be important for sentiment analysis [98].

3.8 Comparing with human-level agreement

To understand how well our DeepMoji classifier performs compared to humans, we created a new dataset of random tweets annotated for sentiment. Each tweet was annotated by a minimum of 10 English-speaking Amazon Mechanical Turkers (MTurk's) living in USA. Tweets were rated on a scale from 1 to 9 with a 'Do not know' option, and guidelines regarding how to rate the tweets were provided to the human raters. The tweets were selected to contain only English text, no mentions and no URL's to make it possible to rate them without any additional contextual information. Tweets where more than half of the evaluators chose 'Do not know' were removed (98 tweets).

Table 3.8: Comparison of agreement between classifiers and the aggregate opinion of Amazon Mechanical Turkers on sentiment prediction of tweets.

	Agreement
Random	50.1%
fastText	71.0%
MTurk	76.1%
DeepMoji	**82.4%**

For each tweet, we select a MTurk rating random to be the 'human evaluation', and average over the remaining nine MTurk ratings are averaged to form the ground truth. The 'sentiment label' for a given tweet is thus defined as the overall consensus among raters (excluding the randomly-selected 'human evaluation' rating). To ensure that the label categories are clearly separated, we removed neutral tweets in the interval [4.5, 5.5] (roughly 29% of the tweets). The remaining dataset consists of 7 347 tweets. Of these tweets, 5000 are used for training/validation and the remaining are used as the test set. Our DeepMoji model is trained using the chain-thaw transfer learning approach.

Table 3.8 shows that the agreement of the random MTurk rater is 76.1%, meaning that the randomly selected rater will agree with the average of the nine other MTurk-ratings of the tweet's polarity 76.1% of the time. Our DeepMoji model achieves 82.4% agreement, which means it is better at capturing the average human sentiment-rating than a single MTurk rater.

Chapter 4

Domain Adaptation through Prediction Propagation

Modern natural language processing (NLP) models rely heavily on machine learning, which has enabled impressive results, but has also made the learned models closely tied with the training data [27]. For NLP, there is enormous amounts of unlabeled data available, and it is thus useful to exploit this data to make the classifiers more robust to data coming from other domains than that of the original training data.

The task of learning models that can exploit unlabeled data sources from other domains to successfully classify observations in those domains has been formalized in [20] under the name of domain adaptation. Many methods have been proposed with some prominent approaches leveraging unsupervised learning to learn a shared representation [7, 95, 15] and others using adversarial loss to learn domain-invariant representations [2, 31, 105].

We find that many of the state-of-the-art methods rely on assumptions about the datasets that are often not upheld for real-world applications. Motivated by these limitations, we propose a new method, Prediction Propagation, that uses the label prediction for each piece of text, together with the surrounding words, to separately reconstruct each word in the text. When backpropagating through the label classifier, the reconstruction loss forces the model to improve its label-related information for domains without labeled data. A new neighborhood encoding architecture and multi-phase training is used to ensure that the label prediction remains a good estimate of the label even when updating the model parameters only using a reconstruction loss. Our method has the desirable properties for real-world modeling and is able to obtain state-of-the-art performance.

4.1 Difficulty of domain adaptation within NLP

Structured correspondence learning (SCL) [7], stacked denoising autoencoders (SDA) [95], and marginalized SDA [15] have all been used for domain adaptation within NLP. These methods first use an unsupervised method to learn a low-dimensional representation and then train a classifier using the source domain labels on this representation. In this way, they rely on the unsupervised projection to a low-dimensional representation being able to capture the information relevant for the label classification, which is a limitation that the current state-of-the-art methods do not have.

A simple approach to learn domain-invariant representations is to use a pretrained deep neural network and align the feature spaces across domains. This approach is used in DAN [55] and Deep CORAL [84]. While these method can be effective, they require that a pretrained network is available. These methods have in the vision domain relied on models that have been pretrained on large-scale annotated datasets such as ImageNet [21], whereas it is not clear in the NLP domain which pretrained models could be used. Similarly, there are other domain adaptation methods that have been used within computer vision, but are not easily applicable to NLP.

The domain-adversarial neural network (DANN) [2, 31] uses an adversarially trained domain classifier to align the feature spaces across domains in addition to a traditional classifier on the source domain. A gradient reversal layer (GRL) ensures that the encoder's parameters are updated such that the domain classification loss increases. While training, the model simultaneously minimizes the label prediction loss while maximizing the domain classifier's cross entropy loss. Many of the recent state-of-the-art methods build on this DANN framework. For instance, [93] proposed the Deep Domain Confusion (Deep DC) method that instead of maximizing the domain classifier's cross-entropy loss minimize the maximum mean discrepancy. Other differences between methods are the model architecture with methods like ADDA and DSN using separate encoders for each domain [94, 8]. The DSN method reconstructs the input as part of the training process, thereby learning which parts of the feature space to share across domains and which to keep private [8]. Similarly, the aspect-augmented adversarial network (AAN) was recently proposed, which stabilizes the adversarial training for NLP applications through the use of an additional word-level autoencoder loss [105].

Our method's reconstruction of individual words in the context of neighboring words has a resemblance to window-based tagging [19] and the continuous bag-of-words (CBOW) method used to learn word vectors [58]. However, to our knowledge, there's no approach that uses the label prediction as part of the text reconstruction in the same way that our method does.

4.2 Linking predictions with text reconstruction

Consider the example in Figure 4-1, where our model is reconstructing the word *sweetiepie* using only the two neighboring words and the text-level label prediction. Given the two neighboring words, a simple model will be able to predict that the reconstructed word should be a noun, but without the text-level label prediction it is unclear if the reconstructed word should be a positive noun (e.g. 'sweetiepie') or a negative noun (e.g. 'moron'). Our method focuses precisely on how to use such reconstruction difficulties to learn label-related information about the target domain.

For the unsupervised domain adaptation task involving a source domain $\mathcal{D}_S$ and a target domain $\mathcal{D}_T$, we aim to learn a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that obtains a low generalization error on the target domain $\mathcal{D}_T$. We have N_S labeled training samples, $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_{N_S}, y_{N_S})\}$, from $\mathcal{D}_S$ and N_T unlabeled samples, $\{(\mathbf{x}_1), ..., (\mathbf{x}_{N_T})\}$, from $\mathcal{D}_T$. The task focus is thus how to make use of the N_T unlabeled training samples to teach the model about $\mathcal{D}_T$ in order to reduce the generalization error.

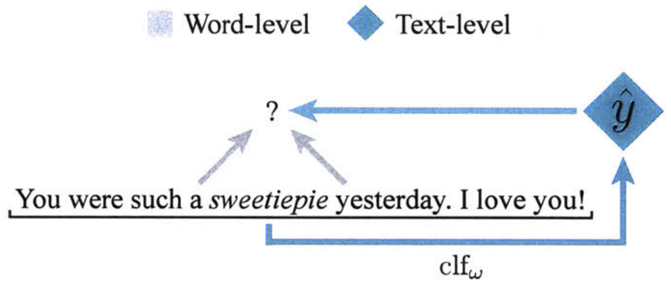

Figure 4-1: Simplified illustration of how our model reconstructs each word using its two neighbors and a text-level prediction of the label.

It is easy to obtain general semantic information about words in $\mathcal{D}_T$ using an unsupervised method like CBOW [58], but it is non-trivial to learn label-related information for each word. For instance, CBOW would likely have difficulty learning that *sweetiepie* is a positive word in the example in Figure 4-1. The goal of our *Prediction Propagation* method is thus to enrich our word embedding function emb($\cdot$) with label-related information about words in $\mathcal{D}_T$ and to enrich our classifier function clf($\cdot$) to account for patterns specific to $\mathcal{D}_T$.

We represent each text of T words as $\mathbf{x} = \{(\mathbf{x}_0), ..., (\mathbf{x}_{T-1})\}$ and assume that the probability of each word $\mathbf{x}_i$ is a function only of its k-adjacent neighbors for some $k > 0$ (i.e. it has the k-Markovian property [32]) and a text-level prediction of the probability distribution over the classes, $\hat{\mathbf{y}}$:

$$p(\mathbf{x}_i|\mathbf{x}_0, ..., \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, ..., \mathbf{x}_{T-1}, \mathbf{y}) = p(\mathbf{x}_i|\mathbf{x}_{i-k}, ..., \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, ..., \mathbf{x}_{i+k}, \hat{\mathbf{y}}) \quad (4.1)$$

The central idea of this method is to use the predicted distribution $\hat{\mathbf{y}}$ as part of a generative model, thereby learning label-related information for $\mathcal{D}_T$ by backpropagating through the classifier. To do this properly requires a special architecture and training procedure that is presented in the remainder of this section.

4.3 Neighborhood encoding architecture

We define a new neighborhood encoding architecture (see Figure 4-2) that is composed of five functions, emb_τ, enc_ϕ, dec_ψ, clf_ω, and glo_δ with respectively τ, ϕ, ψ, ω, and δ as their parameters. An input x to the model is a sentence composed of T words, $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{T-1}$, that are each processed by the embedding function, thereby obtaining a representation $\mathbf{e}_i = \text{emb}_\tau(\mathbf{x}_i)$ for the word at index i. The neighborhood encoding function $\text{enc}_\phi(\mathbf{e}_i)$ then encodes each embedded word $\mathbf{e}_i$ using its neighborhood, which is composed of the k nearest words on either size (but not the word itself), thereby making use of the k-Markovian property previously discussed. The neighborhood encoding of a word is thus $\mathbf{c}_i = \text{enc}_\phi(\mathbf{e}_{i-k}, \ldots, \mathbf{e}_{i-1}, \mathbf{e}_{i+1}, \ldots, \mathbf{e}_{i+k})$.

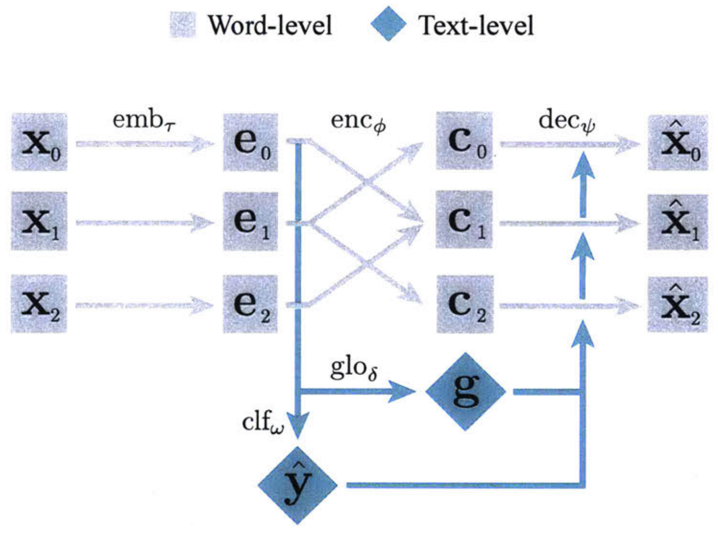

Figure 4-2: Illustration of the architecture for our method with $T = 3$ and $k = 1$. Each word is encoded based on that word's two neighboring words. The predicted distribution over the labels, $\hat{\mathbf{y}}$, is computed for the entire text, which is then concatenated to each encoded word as part of the decoding. Similarly, a global representation $\mathbf{g}$ is computed and concatenated. The decoder has thus access to word-level neighborhood encodings, the text-level prediction, and the text-level global representation.

In parallel to the neighborhood encoding of each word, a classifier function predicts the probability distribution over the labels from the embedding representations, i.e. $\hat{\mathbf{y}} = \text{clf}_\omega(\mathbf{e})$, where $\mathbf{e}$ contains all embedded words $\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_{T-1}$. This text-level prediction is used as part of the input to decoding. Similarly, we compute a global text-level representation $\mathbf{g} = \text{glo}_\delta(\mathbf{e})$ of dimensionality l. The global representation is added to help $\hat{\mathbf{y}}$ remain a good estimate of y by capturing the main global patterns that are relevant for reconstruction, thus preventing the gradients from strongly pushing $\hat{\mathbf{y}}$ towards capturing this information. The benefit of the global representation is confirmed experimentally in Section 4.7.

The decoder function is applied to each word to reconstruct that part of the input, i.e. $\hat{\mathbf{x}}_i = \text{dec}_\psi(\mathbf{c}_i, \hat{\mathbf{y}}, \mathbf{g})$. As opposed to a classic autoencoder [95], the entire input is not compressed to a single vector representation, but rather each word x_i is represented independently by its word-level neighborhood representation c_i, the text-level predicted distribution $\hat{\mathbf{y}}$, and the global representation $\mathbf{g}$. Figure 4-2 illustrates this network architecture for an input of size $T = 3$.

The key to our model's success is the connection from the text-level prediction $\hat{\mathbf{y}}$ to each word as it is being decoded. From an implementation perspective, the prediction is simply repeated with an identical prediction value being concatenated to each encoded word $\mathbf{c}_0, \mathbf{c}_1 \ldots \mathbf{c}_{T-1}$. The same goes for the global representation $\mathbf{g}$. We tie weights ψ with the weights τ for the embedding function, thereby reducing overfitting and the numbers of parameters in the model [42, 68].

We will now describe the layers we use for the four functions of our neighborhood encoding architecture: emb_τ, enc_ϕ, dec_ψ and clf_ω.

The embedding function emb_τ consists of a single embedding layer with an embedding size of $m + n + l$, where $m = 64$ and $l = 8$ are hyperparameters and n is the number of classes. The embedding weights, τ, is consequently a matrix of size $V \times (m + n)$, where V is the vocabulary size.

The classifier function clf_ω has an attention layer that averages the representations across words by weighing each word based on their attention importance score as done in [28] and a bidirectional long short-term memory [39, 76] module with 512 dimensions in each direction for detecting sequential patterns. These two layers are combined using a fully-connected layers with 1024 units and a Softmax layer that computes a probability distribution over the n classes. The global representation function glo_δ uses the same layers as the classifier function, but has a final layer of dimensionality $l = 8$ with a tangents hyperbolic activation rather than the Softmax layer.

The neighborhood encoding function enc_ϕ uses a *center-masked* convolutional layer with a filter size of 3 to encode each word using its two neighboring words. We define a center-masked convolution as a standard convolution that is applied across a feature map, where the central neuron is masked by multiplying it with zero to have no influence on the filters (see enc_ϕ in Figure 4-2). The center-masked convolutional layer has 2048 filters and is followed by a time-distributed fully-connected layer that projects each word independently down to 64 dimensions. Both layers use the ReLU activation.

The decoder function multiplies the concatenated input from enc_ϕ and clf_ω with a matrix of size $(m+n) \times V$. A bias is added for each word and the softmax activation function is applied, thereby for each word producing a probability distribution over all the words in the vocabulary.

4.4 Multi-phase training to learn desired representations

In order to learn label-related information about words and patterns occurring in $\mathcal{D}_T$ by backpropagating the gradients from a reconstruction loss through the classifier, it is crucial that $\hat{\mathbf{y}}$ remains a good estimate of y throughout training. This is non-trivial as the gradients from the reconstruction loss will attempt to move the prediction $\hat{\mathbf{y}}$ away from being a good estimate of y and instead towards something more useful for reconstructing the words. One could imagine trying to make $\hat{\mathbf{y}}$ remain a good estimate by using a classification loss based on the labeled samples from $\mathcal{D}_S$, but this would introduce a dataset-dependent hyperparameter related to the trade-off between the reconstruction loss and classification loss.

We ensure that $\hat{\mathbf{y}}$ remains a good estimate by training the model in five consecutive phases. Each phase involves minimizing one or more of the model's three losses with respect to a subset of the parameters τ, ϕ, ψ and ω. The three losses are the classification loss on the source dataset, $\mathcal{L}_{clf}$, and the two reconstructions losses, $\mathcal{L}_{source}$ and $\mathcal{L}_{target}$, making the total training loss:

$$\mathcal{L} = \lambda_{clf}\,\mathcal{L}_{clf} + \lambda_{source}\,\mathcal{L}_{source} + \lambda_{target}\,\mathcal{L}_{target} \tag{4.2}$$

While this training scheme may seem excessive, it is unlike most other domain adaptation methods free of hyperparameters that need to be tuned for each new dataset. Moreover, the main difference between phases is which parameters are trainable and which losses are used, making the training easy to implement in any modern deep learning library. The loss multipliers, λ_{clf}, λ_{source} and λ_{target} are always binary, meaning that the loss is simply on or off for any given phase. Table 4.1 details which parameters of the model are trainable in which phase and which losses are used. The first three phases teaches the model how to reconstruct the words $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{T-1}$ and predict $\hat{\mathbf{y}}$ for $\mathcal{D}_S$, whereas the last two phases make use of the trained model to learn label-related information for $\mathcal{D}_T$. In particular, the fourth phase updates the word embeddings parameters τ to learn the meanings of previously unknown words and new meanings of existing words. Similarly, the fifth phase updates ω to capture patterns specific to $\mathcal{D}_T$. As phase 1 does not yet have a trained classifier to provide $\hat{\mathbf{y}}$, the labels are provided instead. Table 4.7 in Section 4.7 show how the model is improving in each phase.

Table 4.1: Parameters being updated in each phase using which losses. Loss multipliers are binary.

Phase	Parameters	λ_{clf}	λ_{source}	λ_{target}
1	τ, ϕ, ψ, δ	0	1	0
2	ω	1	0	0
3	τ, ϕ, ψ, δ	0	1	0
4	τ, ϕ, ψ, δ	0	1	1
5	ω	0	1	1

The training is done using the Adam optimizer [48] using the default parameters and with gradient clipping of the norm set at 1. In all phases the model is trained until convergence as determined by lack of improvement on the validation set, which contains observations from $\mathcal{D}_S$ and $\mathcal{D}_T$ depending on the losses in that phase.

4.5 Desirable properties for domain adaptation methods

Due to the many ways of designing domain adaptation methods, we aim at identifying some desirable properties for these methods that can put our method in the context of previous work. We identify three such desirable properties.

Table 4.2: Properties of various NLP domain adaptation methods.

Method	Learn w/o pretrained params	Classify w/o domain info	Train on different label distributions
DAN [55]	no	yes	yes
Deep CORAL [84]	no	yes	yes
ADDA [94]	yes	no	no
DSN [8]	yes	no	suboptimal[1]
Deep DC [93]	yes	yes	no
DANN [2]	yes	yes	no
AAN [105]	yes	yes	no
Prediction Propagation (ours)	yes	yes	yes

First, it is desirable that the method can be trained from scratch on the dataset without any pretrained parameters because there are many important real-world tasks for which there are no pretrained classifiers available to adapt. While most of the state-of-the-art domain adaptation methods are able to do this, the DAN [55] and Deep CORAL [84] rely on aligning features of pretrained neural network models and are thus restricted in the tasks they can be applied to.

Secondly, it is desirable that the method can classify an observation without knowledge of the domain from which that observation originated. Some domain adaptation

[1]DSN's similarity loss is defined as the adversarial loss from either DANN or Deep DC, both of which have difficulty training on domains with different label distributions. Table 3 in the DSN paper shows that the model works without this loss, but achieves results below state of the art for 2 out of 4 benchmark datasets.

methods require such information for each observation due to e.g. the use of separate encoders for different domains [8, 94]. While this issue could potentially solved by having a domain classifier that is used as part of the preprocessing, it is not always trivial to determine the domain of individual observations in NLP. Additionally, for real-world applications some new observations may occur that cannot be clearly defined as being part of any of the existing domains used for training. It is not clear how these methods with separate layers for different domains would handle such observations.

Thirdly, it is desirable that a method is able to model domains with different label distributions as many real-world domains vary in their label distributions. For instance, Amazon reviews have a proportion of positive reviews ranging from 78% to 93% depending on the product category (see the supplementary material). All the methods that are rooted in the gradient reversal technique from DANN (i.e. ADDA, DSN, Deep DC, DANN, AAN) suffer from an inability to handle such differences in label distributions across domains. For these methods, the domain classifier exploits the label information encoded in the feature vector to predict the domain, causing the reversed gradient to remove information pertinent to the classification task. This issue is formalized in the supplementary information and experimentally confirmed in Section 4.6.

Table 4.2 summarizes the properties of the state-of-the-art methods within NLP (see Section 4.1 for other methods). Prediction Propagation stands out as the only method with all three desirable properties. With this paper's focus on domain adaptation for real-world applications, the DANN, Deep DC and AAN methods are the most relevant baselines due to their ability to learn without pretrained parameters and classify observations without domain information. Recent work has shown that DANN performs comparable or better than Deep DC [94, 56], which is why we only use DANN and AAN as baseline comparisons in Section 4.6.

4.6 Benchmarking and the issue of unbalanced datasets

As described in Section 4.5 we compare our model to DANN and AAN. For our implementation of DANN we use similar embedding and classification layers as for our own model, while using the same gradient reversal factor schedule as in the original DANN paper (see the supplementary material for implementation details). For AAN we use the authors' GitHub code and default hyperparameters.

We create three new datasets due to lack of proper evaluation datasets for domain adaptation in NLP. An overview of the datasets can be seen in Table 4.3. For all datasets we use 5000 observations for the validation set and 15000 observations for the test set. To tokenize the texts and create the vocabularies we use the code from DeepMoji [28].

Table 4.3: Description of benchmark datasets.

Experiment	Study	Task	Classes	N_{train}	N_{val}	N_{test}	Bal	Unbal
Twitter $\leftrightarrow$ Amazon	[91, 37]	Sentiment	2	200000	5000	15000	✓	
Amazon Categories	[37]	Sentiment	2	40000	5000	15000	✓	✓
AG News $\leftrightarrow$ Yahoo	[103]	Topic	2	40000	5000	15000	✓	

For sentiment classification we consider domain adaptation between Amazon reviews and tweets on Twitter as well as between three Amazon product categories. Previous work has also used Amazon product categories to evaluate domain adaptation [31, 15], but the review datasets used in these previous papers only had 2000 labeled source samples, making the dataset small for modern standards. We thus create our own binary classification Amazon review datasets based on previously released data [37]. Our datasets are substantially larger, making for a more realistic test case. The reviews are on an ordinal scale from 1 to 5, which we binarize by regarding reviews with scores below 3 as negative and above 3 as positive. Reviews with a score of 3 are discarded to establish a clear separation between negative and positive reviews. To make for a more challenging domain adaptation task, we use only reviews with less than 30 tokens and remove all *trivially easy* observations. These observations are defined as the ones that a bag-of-words logistic regression using only a vocabulary of the top 1000 tokens predicts with 99% confidence or more. In addition to creating balanced datasets, we also randomly sample observations from the Amazon categories to create a version of that dataset that is naturally unbalanced, varying from 78% to 93% of the reviews being positive depending on the product category (see the supplementary material for details).

In agreement with prior work [61, 87], we define positive sentiment for tweets as those containing a positive emoji and, similarly, negative sentiment for those containing a negative emoji. To handle the noisy text on Twitter we use the tokenization scheme and vocabulary from DeepMoji [28], where words with 2 or more repeated characters are shortened to use the same token. Furtermore, all URLs, numbers, and @-mentions are replaced by special tokens.

For topic classifciation we consider domain adaptation between between Yahoo Answers and AG News [103] where the two overlapping topics in both domains, Science/Technology and Sports, are used as our two prediction classes. Both corpora are balanced. A large proportion of the observations had the website domain as part of the text, making it trivially easy to classify these observation (e.g. if the domain is www.spacescience.com). We thus remove the website domains from the texts as part of the preprocessing.

To evaluate the performance on the datasets we use the area under the curve (AUC) of the receiver operating characteristic, which is suited for evaluating both balanced and unbalanced datasets. Table 4.4 shows that Prediction Propagation is the method of the three that achieves state-of-the-art performance across the most datasets. All the results are averaged across 3 runs. Additionally, we compute the standard error of the mean (SEM) of the AUC for each combination of method and dataset.

To our knowledge, DANN and AAN have only been benchmarked on domain adaptation tasks, where the distribution of labels is uniform for both the source and

Table 4.4: Balanced tasks. AUC on the target dataset averaged across 3 runs. Higher is better.

	DANN	AAN	Our method
Amazon → Twitter	.734 ± .006	.752 ± .032	**.766 ± .004**
Twitter → Amazon	.822 ± .008	.851 ± .015	**.874 ± .016**
AG → Yahoo	.773 ± .026	**.913 ± .003**	.894 ± .004
Yahoo → AG	.948 ± .007	.965 ± .002	**.978 ± .003**
Books → Movies	**.947 ± .001**	.898 ± .004	.938 ± .001
Books → Clothing	.911 ± .004	.876 ± .005	**.918 ± .001**
Movies → Books	**.929 ± .002**	.903 ± .001	.924 ± .001
Movies → Clothing	**.918 ± .003**	.888 ± .012	.899 ± .002
Clothing → Books	.842 ± .014	.864 ± .007	**.900 ± .002**
Clothing → Movies	.896 ± .005	.846 ± .017	**.921 ± .003**

Table 4.5: Unbalanced tasks. AUC on the target dataset averaged across 3 runs. Higher is better.

	DANN	AAN	Our method
Books → Movies	.848 ± .008	.680 ± .067	**.918 ± .005**
Books → Clothing	.836 ± .003	.732 ± .061	**.901 ± .001**
Movies → Books	.816 ± .003	.614 ± .021	**.912 ± .006**
Movies → Clothing	.819 ± .004	.786 ± .023	**.898 ± .005**
Clothing → Books	.724 ± .008	.693 ± .077	**.848 ± .005**
Clothing → Movies	.743 ± .010	.738 ± .033	**.870 ± .003**

target domain. However, obtaining a balanced target dataset necessarily requires access to the labels, which would not be present in a real-world application of domain adaptation. We thus evaluate the methods, where the source domain is balanced (as often done for training) and the target domain is naturally unbalanced, i.e. the observations are sampled randomly from the original dataset. Table 4.5 shows that DANN and AAN perform substantially worse on these tasks with a unbalanced target domain, thereby confirming the issue discussed in Section 4.5 and formalized in the supplementary material. Our method substantially outperforms DANN and AAN on these unbalanced datasets.

4.7 Analyzing the modeling choices

In this section we analyze the modeling choices we've made. Firstly, we analyze our underlying premise of the backpropagation through the classifier being crucial for the performance of our model. We run the balanced Amazon $\rightarrow$ Twitter experiment, where we stop the gradient from backpropagating from the decoder to the classifier. The AUC on the target domain validation set drops from .766 to .731, which is below the current state-of-the-art methods, thereby emphasizing the importance of propagating the predictions from the decoder through the classifier.

Our method does not have any training hyperparameters, but it has two hyperparameters related to the architecture: neighborhood size k and global dimensionality l. Table 4.6 shows a grid search over hyperparameter values with the results being averages across 3 runs. We find that $k = 3$ and $l = 8$ provide the best results and we use these values across all the experiments.

Table 4.6: Grid search over the neighborhood size k and dimensionality of the global representation l. The values are AUC for the task Amazon $\rightarrow$ Twitter.

	l			
k	0	4	8	16
1	.747	.756	.731	.732
3	.753	.741	**.766**	.732
5	.753	.756	.732	.760

Table 4.7: Accuracies for sentiment prediction and word reconstruction on the validation set for target domain for the balanced task Amazon $\rightarrow$ Twitter.

Phase	Sentiment	Words
1	–	$14.5 \pm .1$
2	$61.8 \pm .4$	–
3	$62.8 \pm .5$	$14.5 \pm .2$
4	$68.1 \pm .1$	$29.1 \pm .6$
5	$68.5 \pm .2$	$29.2 \pm .3$

Lastly, to understand the impact of each of our training phases we evaluate the sentiment and word reconstruction accuracies on the target domain for the balanced task Amazon $\rightarrow$ Twitter. As seen in Table 4.7 each phase improves the performance. Phase 4 substantially improves the word reconstruction accuracy, which is intuitive as this is the first phase, where the target domain reconstruction loss, $\mathcal{L}_{target}$ is used. Phase 1 and 4 take the longest with the remaining three phases accounting for less than 25% of the overall training time.

Chapter 5

Disentangling Content and Style using Representation Distillation

It is trivial for a computer to express specific emotional content if given full flexibility over the content of the text. However, that is not desirable as the communication often has a purpose, requiring the computer to communicate certain content. The challenge thus lies in communicating the same content while changing the style of the text to express specific emotions, which we refer to as 'style modification'. A simple approach to modify the emotional style of a text is to replace sentiment-bearing words with antonyms, but this does not capture the full richness of emotional content in language, which is why recent research has attempted style modification of text using machine learning [78, 47]. However, the existing methods have important limitations wrt. content consistency and flexibility of style control. We propose a new approach that learns disentangled representations of style and style using a specialized dilated convolutional autoencoder architecture that uses a pretrained classifier and adversarial training. In addition to allowing style modification, this method also enables style-independent content representations that may be useful for downstream tasks.

5.1 Usefulness of disentangled representations

Decomposing an input into disentangled representations of style and content has attracted a lot of attention within the machine learning community [88, 51, 16, 57]. In the context of handwriting recognition, the style can be the font [88], whereas for language the style can include factors such as sentiment and formality. It is often desirable to be able to perform style modification, where the style is changed while keeping the content constant. A particular kind of style modification, 'style transfer', entails that the style of an input, x_s is applied to another input, x_t, thereby changing the style of x_t. One way to perform style transfer is through the use of disentangled representations, where style and content each have a representation that is independent of the other.

5.2 Style transfer with discrete words

A lot of previous work in NLP has attempted to achieve disentangled representations by modeling the data in the VAE framework [49], thereby placing a strict prior on the code space [9, 77, 41] that also allows them to sample sentences. One way to enforce that the style and content representations are independent in a VAE model is through the use of separate discriminators to classify these representations [41]. However, using this approach for style transfer does experience failure cases with the content of the reconstructed output, $\hat{x_t}$, being very different from the content of x_t.

Two recent papers model the problem of style transfer more explicitly [78, 47]. In one of them the style transfer task is treated as a translation problem with two encoders and decoders [78] with the encoders and decoders being cross-aligned using Professor Forcing [54]. The other paper regularizes the style and the content to be independent [47] through the use of adversarial training framework originally designed for domain adaptation [31] and a continuous-space generator based on the Wasserstein distance [3]. These two papers represent the current the state-of-the-art for style transfer on text and will be the main comparisons when evaluating our approach.

A major difficulty for style transfer on text is the ability to modify the style while preserving the content, partly due to the discrete nature of the data. The two methods discussed above also suffer from this issue, thus making some transferred sentences be very different from the original ones (see Table 6 in [47]).

5.3 Disentangled representations through distillation

To determine which factors of variation from the input should be captured in the style representation, previous work made use of labeled datasets to train a classifier as part of the neural network [41, 47]. Training a classifier as part of the style transfer model requires the network to simultaneously learn the style-specific factors and the reconstruction of the input, making the optimization task more difficult. Moreover, the relationship between style and content can be highly complex, thus requiring a large labeled dataset to properly capture these dynamics.

We prevent these issues altogether by using a pretrained domain-invariant classifier[1] to guide our model. Similar to how model distillation has been used for training small students models from larger teacher models [38], we distill the knowledge of the classifier to obtain disentangled representations. Accordingly, we name our method *Representation Distillation (RepDistil)*. Having the pretrained classifier as an integral part of the architecture, allows it to provide gradients for the decoder, thereby guiding it to how it can achieve a desired style. In this way, the pretrained classifier eases the optimization task and allows our model to be trained on unlabeled datasets.

To express content with a specific emotional style it is helpful decompose an input x into two separate representations, c and s, containing respectively information about the content and the style. While this can be accomplished in a completely unsupervised manner for simple styles [16, 69], language is so rich in variations that it seems infeasible for more complex styles. Moreover, completely unsupervised learn-

[1] If one such classifier does not exist for a specific labeled dataset, it is easy to create one by first training a classifier before training our style transfer network.

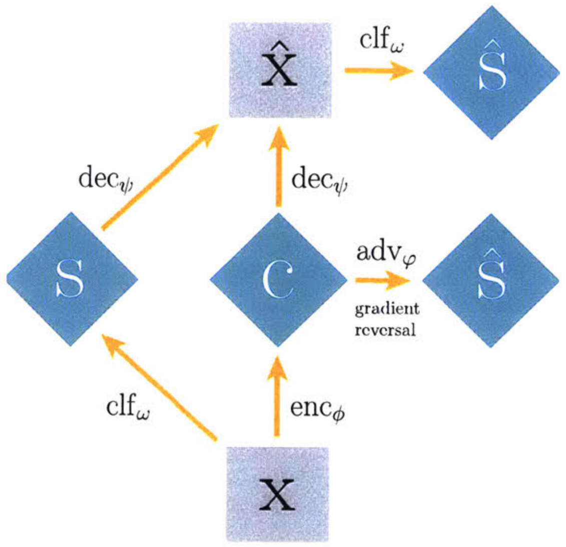

Figure 5-1: High-level illustration of the proposed architecture for learning disentangled representations and performing style modification. Grey boxes are the observed texts (before and after modification) while blue diamonds are representations computed by the model.

Algorithm 1 Training

```
while not converged do
    Compute $\hat{s} = \text{adv}_\varphi(\text{enc}_\phi(x))$. Update $\varphi$. Repeat $k$ times.
    Compute $\hat{x} = \text{dec}_\psi(\text{enc}_\phi(x), \text{clf}_\omega(x))$, $\hat{s} = \text{adv}_\varphi(GRL(\text{enc}_\phi(x)))$. Update $\phi$, $\psi$.
    Compute $\hat{s} = \text{clf}_\omega(\text{dec}_\psi(\text{enc}_\phi(x), \text{clf}_\omega(x)))$. Update $\psi$.
```

ing makes the trained model hard to use for style modification as different training sessions will likely yield different factors of variation as the detected style factors. Consequently, we examine the approach, where additional information is available on which factors of style should be separated into s. In particular, we make use of a pretrained classifier that maps the input to a style representation using a parameterized function $s = \text{clf}_\omega(x)$ with parameters ω.

Using the pretrained classifier allows us to get the style representation, s, but it is not trivial to obtain a content representation, c, that is independent of the style. To obtain these disentangled representation we use an extension of the classic autoencoder network architecture [95]. An autoencoder consists of two parts. An encoder maps the input, x, into a content representation, c, using a parameterized function

$c = \text{enc}_\phi(x)$ with parameters ϕ. The content representation is then mapped back to the original space by a decoder, $\hat{x} = \text{dec}_\psi(c)$ with parameters ψ. The autoencoder is perfectly reconstructing x when $x = \hat{x}$. If $dim(c) = dim(x)$, then a trivial solution for the autoencoder is to learn the identity function such that $\text{enc}_\phi(x) = x$ and $\text{dec}_\psi(c) = x$. Instead, $dim(c)$ is chosen to be a low number such that the autoencoder attempts to learn a lower dimensional manifold that contains all of the information in the input space and thereby compresses the input. The autoencoder thus models the data as $p(x|c)$, where $c = \text{enc}_\phi(x)$ is a low-dimensional representation of the input.

5.4 Adversarial training of the RepDistill model

A naïve approach for controlling the style of the reconstruction is to model $p(x|c, s)$ instead, where the style representation, s, is concatenated to the content representation. However, as all of the information needed for the reconstruction is available in x, the decoder often learns to ignore the style variable. If this happens, it becomes impossible to achieve different reconstructions by modifying the style representation.

To enforce c to not contain information about s, we make use of adversarial training [34] and the gradient reversal layer (GRL) [31]. This layer acts as the identity function on the forward pass, but reverses the gradient during backpropagation by multiplying it with -1. In this way, any loss coming through the GRL will optimize parameters earlier in the model to make the loss worse. An adversarial classifier with parameters φ is trained to predict the style representation through a GRL layer, i.e. $\hat{s} = \text{adv}_\varphi(GRL(c))$. During this adversarial training, the φ parameters will thus be updated to better be able to predict s, while the parameters of the encoder, ϕ, will be updated to make c contain less information useful for predicting s. Consequently, properly reconstructing x requires the use of the concatenated style representation, thereby ensuring that modifying s changes $\hat{x}$.

The content and style might be related in complex ways, which can make it difficult for the decoder to correctly reconstruct the style. In practice, the decoder may choose to partially ignore reconstructing the style in favor of better modeling the content.

This issue can be mitigated by having an extra loss for the autoencoder's ability to reconstruct the style, thus allowing us to tune the importance of the reconstructed output having the correct style. We implement this loss using the pretrained classifier such that the model attempts to make $\mathrm{clf}_\omega(\hat{x})$ as close as possible to $\mathrm{clf}_\omega(x)$. In this way, the pretrained classifier provides gradients that guides the decoder to achieve a specific style, even on unlabeled datasets. This approach is similar in vein to model distillation, where one or more models are used to train a new smaller model with comparable performance [11, 38]. Accordingly, we find that matching the logits with a mean squared error loss as proposed in [11] for model distillation works well for our approach[2].

An illustration of our neural network architecture can be seen in Figure 5-1. Training the network involves three separate backpropagation steps to ensure that the gradients encourage the encoder to make c independent of s and to make the decoder use s. Algorithm 1 describes the training in more detail. Once trained, the user simply provides an input and a desired style representation to obtain a modified version of the input. Style transfer from an input x_s to another input x_t can therefore be achieved simply by reconstructing x_t with $s = \mathrm{clf}_\omega(x_s)$.

Our network architecture encodes the input into a variable-size content representation that is resolution preserving [45], i.e. scaling the number of hidden states one-to-one with the size of the input. Each word is encoded into its own hidden state vector, which is made available to the decoder. In this way, the variable-size encoding makes the learning task much easier as it allows the decoder direct access to the individual tokens in the input text, thereby allowing the network to model the change in style for a single word directly rather than first having to memorize the entire text into a single vector. Furthermore, this architecture simplifies the optimization task as it makes the paths of the gradients from the output tokens to the corresponding input tokens much shorter. This approach is in contrast to previous work that all encode the content into a single fixed-size vector. The style representation is run through a

[2]The style representation is always kept in logits, incl. the target for the adversarial network and the input to the decoder.

1-layer MLP and concatenated to every time step.

5.5 Evaluating the style transfer performance

At the time of writing this thesis, the RepDistil project is still work in progress, leaving much to be wanted in this evaluation section. Furthermore, the comparison with existing algorithms is complicated by the lack of a single clear metric for evaluating style transfer. Individual papers use different metrics and many use human evaluations without specifying complete experiment design and participant instructions.

The main difficulty of evaluating style transfer models for text is that there will always be a trade-off between thc degree to which the original content remains and the degree to which the style has been transferred. I am currently involved in another research project on identifying better ways of quantifying the performance of style transfer models for text. This part of the RepDistil project has thus purposely not been pursued, but is waiting on new standardized metrics and benchmark datasets from the other project for better comparisons.

Chapter 6

Discussion

This thesis focused on the hard task of teaching machines about emotions, but do the methods proposed here actually take us any closer?

DeepMoji is helping researchers and practitioners that would like to go beyond measuring positive/negative sentiment and instead model various kinds of emotions. The pretrained neural network has been deployed in multiple industry applications (incl. chatbots), thereby helping the machines better understand the emotion content of how we express ourselves. Equally important, DeepMoji is helping open up new avenues of research for scientists interested in understanding phenomena related to emotions. For instance, DeepMoji has been used to model racism [46] and to help us get a better understanding of theory behind emotions [13].

Our Prediction Propagation method and the new benchmark datasets we provide can hopefully help push the field of domain adaptation for NLP forward. This is crucial as NLP methods are notoriously brittle [27]. While the description of the method here focused on classifying emotional content, the method could potentially also benefit a variety of other NLP tasks.

Learning to obtain disentangled representations of messy real-world data could be an important step towards the causal models that some argue are necessary for human-like AI [53]. Due to its discrete nature and long-tailed distribution of words, language is arguably a hard domain for learning disentangled representations.

RepDistil could help solve this problem, although it still needs more work for us to prove that the algorithm works better than the existing methods.

It is clear that none of the methods proposed here will take us all the way to the big vision of an emotional AI. Nevertheless, I do believe that the methods proposed in this thesis help move the field in the right direction.

Chapter 7

Conclusion

This thesis explored how one could design algorithms for modeling the rich spectrum of emotions that humans employ. I have done that through three research projects on this overall topic. In the first paper, we show how the billions of texts on social media with emojis can be used for training deep learning models that obtain state-of-the-art performance across various emotion-related NLP tasks. Our DeepMoji model has been deployed in multiple industry applications and is being used in academia to analyze phenomena related to emotions. The second paper shows how the current state-of-the-art domain adaptation methods lack one or more desirable properties for real-world NLP applications. Motivated by these limitations we have introduced the Prediction Propagation method, which has these desirable properties and is able to obtain state-of-the-art performance. Lastly, the third paper proposes the RepDistill method for disentangling content and style of texts. Although additionally work is required to verify this method's efficacy, the initial results are promising. All in all, there's still a long way to go before we reach machines that actually understand emotions, but the methods proposed in this thesis move us in the right direction while having practical applicability for both industry and academia.

Bibliography

[1] Hyungil Ahn and Rosalind W Picard. Affective-cognitive learning and decision making: A motivational reward framework for affective agents. In *International Conference on Affective Computing and Intelligent Interaction*, pages 866–873. Springer, 2005.

[2] Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-adversarial neural networks. In *Second Workshop on Transfer and Multi-Task Learning: Theory meets Practice (NIPS 2014)*, 2014.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations (ICLR)*, 2014.

[5] Lisa Feldman Barrett. The conceptual act theory: A précis. *Emotion Review*, 6(4):292–297, 2014.

[6] Yoshua Bengio et al. Deep learning of representations for unsupervised and transfer learning. In *29th International Conference on Machine learning (ICML) – Workshop on Unsupervised and Transfer Learning*, volume 27, pages 17–36, 2012.

[7] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2006.

[8] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[9] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[10] Margaret M Bradley and Peter J Lang. Affective norms for english words (anew): Instruction manual and affective ratings. Technical report, Citeseer, 1999.

[11] Cristian BuciluÇÕ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM, 2006.

[12] Sven Buechel and Udo Hahn. Emotion analysis as a regression problem - dimensional models and their implications on emotion representation and metrical evaluation. In *22nd European Conference on Artificial Intelligence (ECAI)*, 2016.

[13] Andres Campero, Bjarke Felbo, Joshua B Tenenbaum, and Rebecca Saxe. A first step in combining cognitive event features and natural language representations to predict emotions. *arXiv preprint arXiv:1710.08048*, 2017.

[14] Ginevra Castellano, Loic Kessous, and George Caridakis. Emotion recognition through multiple modalities: face, body gesture, speech. In *Affect and emotion in human-computer interaction*, pages 92–103. Springer, 2008.

[15] Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *International Conference on Machine Learning (ICML)*, 2012.

[16] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[17] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[18] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *25th International Conference on Machine learning (ICML)*, pages 160–167, 2008.

[19] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research (JMLR)*, 2011.

[20] Hal Daume III and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 2006.

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[22] Jan Deriu, Maurice Gonzenbach, Fatih Uzdilli, Aurelien Lucchi, Valeria De Luca, and Martin Jaggi. Swisscheese at semeval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision. *Proceedings of SemEval*, pages 1124–1128, 2016.

[23] Karthik Dinakar, Roi Reichart, and Henry Lieberman. Modeling the detection of textual cyberbullying. *The Social Mobile Web*, 11(02):11–17, 2011.

[24] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *31th International Conference on Machine Learning (ICML)*, volume 32, pages 647–655, 2014.

[25] Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. emoji2vec: Learning emoji representations from their description. In *4th International Workshop on Natural Language Processing for Social Media (SocialNLP)*, 2016.

[26] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research (JMLR)*, 11:625–660, 2010.

[27] Allyson Ettinger, Sudha Rao, Hal Daumé III, and Emily M Bender. Towards linguistically generalizable nlp systems: A workshop and shared task. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[28] Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[29] Sandra Clara Gadanho and John Hallam. Robot learning driven by emotions. *Adaptive Behavior*, 9(1):42–64, 2001.

[30] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *30th Conference on Neural Information Processing Systems (NIPS)*, pages 1019–1027, 2016.

[31] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research (JMLR)*, 2016.

[32] Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice.* CRC press, 1995.

[33] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12), 2009.

[34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[35] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[36] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[37] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *International Conference on World Wide Web (WWW)*, 2016.

[38] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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

[40] Xia Hu, Jiliang Tang, Huiji Gao, and Huan Liu. Unsupervised sentiment analysis with emotional signals. In *Proceedings of the 22nd international conference on World Wide Web (WWW)*, pages 607–618. ACM, 2013.

[41] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596, 2017.

[42] Hakan Inan, Khashayar Khosravi, and Richard Socher. Tying word vectors and word classifiers: A loss framework for language modeling. In *International Conference on Learning Representations (ICLR)*, 2017.

[43] Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. Are word embedding-based features useful for sarcasm detection? In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[44] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[45] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016.

[46] Richard Kim, Bjarke Felbo, Nick Obradovich, and Iyad Rahwan. Quantifying structural factors affecting racial tensions during 2016 us presidential election. In *International Conference on Computational Social Science (IC2S2)*, 2017.

[47] Yoon Kim, Kelly Zhang, Alexander M Rush, Yann LeCun, et al. Adversarially regularized autoencoders for generating discrete structures. *arXiv preprint arXiv:1706.04223*, 2017.

[48] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015.

[49] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, pages 1097–1105, 2012.

[51] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547, 2015.

[52] FA Kunneman, CC Liebrecht, and APJ van den Bosch. The (un)predictability of emotional hashtags in twitter. In *52th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 2014.

[53] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

[54] Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609, 2016.

[55] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning (ICML)*, 2015.

[56] Zelun Luo, Yuliang Zou, Judy Hoffman, and Li F Fei-Fei. Label efficient learning of transferable representations acrosss domains and tasks. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[57] Michael F Mathieu, Junbo Jake Zhao, Junbo Zhao, Aditya Ramesh, Pablo Sprechmann, and Yann LeCun. Disentangling factors of variation in deep representation using adversarial training. In *Advances in Neural Information Processing Systems*, pages 5040–5048, 2016.

[58] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *27th Conference on Neural Information Processing Systems (NIPS)*, pages 3111–3119, 2013.

[59] Saif Mohammad. #emotional tweets. In *The First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 246–255. Association for Computational Linguistics, 7-8 June 2012.

[60] Agnes Moors. Flavors of appraisal theories of emotion. *Emotion Review*, 6(4): 303–307, 2014.

[61] Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. Semeval-2016 task 4: Sentiment analysis in twitter. In *10th International Workshop on Semantic Evaluation (SemEval)*, pages 1–18, 2016.

[62] Finn Årup Nielsen. A new anew: Evaluation of a word list for sentiment analysis in microblogs. In *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages*, 2011.

[63] Michael A. Nielsen. *Neural Networks and Deep Learning.* Determination Press, 2015.

[64] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision (ECCV)*, pages 69–84. Springer, 2016.

[65] Shereen Oraby, Vrindavan Harrison, Lena Reed, Ernesto Hernandez, Ellen Riloff, and Marilyn Walker. Creating and characterizing a diverse corpus of sarcasm in dialogue. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, page 31, 2016.

[66] Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135, 2008.

[67] Rosalind W Picard. Toward computers that recognize and respond to user emotion. *IBM systems journal*, 39(3.4):705–719, 2000.

[68] Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2017.

[69] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

[70] Pramila Rani, Changchun Liu, Nilanjan Sarkar, and Eric Vanman. An empirical study of machine learning techniques for affect recognition in human–robot interaction. *Pattern Analysis and Applications*, 9(1):58–69, 2006.

[71] Jonathon Read. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *ACL student research workshop*, pages 43–48. Association for Computational Linguistics, 2005.

[72] Sara Rosenthal, Noura Farra, and Preslav Nakov. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518, 2017.

[73] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive Modeling*, 5, 1988.

[74] Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. Evaluation datasets for twitter sentiment analysis: a survey and a new dataset, the sts-gold. In *Workshop: Emotion and Sentiment in Social and Expressive Media: approaches and perspectives from AI (ESSEM) at AI*IA Conference*, 2013.

[75] Rebecca Saxe and Laura Schulz. 9.s913 graduate seminar on emotion, 2017.

[76] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 1997.

[77] Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. A hybrid convolutional variational autoencoder for text generation. *arXiv preprint arXiv:1702.02390*, 2017.

[78] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. *arXiv preprint arXiv:1705.09655*, 2017.

[79] Valentina Sintsova, Claudiu-Cristian Musat, and Pearl Pu. Fine-grained emotion recognition in olympic tweets based on human computation. In *4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA)*, 2013.

[80] Jonas Sjöberg and Lennart Ljung. Overtraining, regularization and searching for a minimum, with application to neural networks. *International Journal of Control*, 62(6):1391–1407, 1995.

[81] Jacopo Staiano and Marco Guerini. Depechemood: A lexicon for emotion analysis from crowd-annotated news. In *52th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 2014.

[82] Carlo Strapparava and Rada Mihalcea. Semeval-2007 task 14: Affective text. In *4th International Workshop on Semantic Evaluations (SemEval)*, pages 70–74. Association for Computational Linguistics, 2007.

[83] Yoshihiko Suhara, Yinzhan Xu, and Alex'Sandy' Pentland. Deepmood: Forecasting depressed mood based on self-reported histories via recurrent neural networks. In *Proceedings of the 26th International Conference on World Wide Web*, pages 715–724. International World Wide Web Conferences Steering Committee, 2017.

[84] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision (ECCV) 2016 Workshops*, 2016.

[85] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *28th Conference on Neural Information Processing Systems (NIPS)*, pages 3104–3112, 2014.

[86] Jared Suttles and Nancy Ide. Distant supervision for emotion classification with discrete binary values. In *International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, pages 121–136. Springer, 2013.

[87] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *52th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1555–1565, 2014.

[88] Joshua B Tenenbaum and William T Freeman. Separating style and content. In *Advances in neural information processing systems*, pages 662–668, 1997.

[89] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`.

[90] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12):2544–2558, 2010.

[91] Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. Sentiment strength detection for the social web. *Journal of the American Society for Information Science and Technology (JASIST)*, 63(1):163–173, 2012.

[92] Jessica L Tracy. An evolutionary approach to understanding distinct emotions. *Emotion Review*, 6(4):308–312, 2014.

[93] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *International Conference in Computer Vision (ICCV)*, 2015.

[94] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[95] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[96] Marilyn A Walker, Jean E Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. A corpus for research on deliberation and debate. In *International Conference on Language Resources and Evaluation (LREC)*, pages 812–817, 2012.

[97] Harald G Wallbott and Klaus R Scherer. How universal and specific is emotional experience? evidence from 27 countries on five continents. *International Social Science Council*, 25(4):763–795, 1986.

[98] Michael Wiegand, Alexandra Balahur, Benjamin Roth, Dietrich Klakow, and Andrés Montoyo. A survey on the role of negation in sentiment analysis. In *Workshop on Negation and Speculation in Natural Language Processing (NeSp-NLP)*, pages 60–68. Association for Computational Linguistics, 2010.

[99] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning (ICML)*, pages 2048–2057, 2015.

[100] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. Hierarchical attention networks for document classification. In *HLT-NAACL*, 2016.

[101] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[102] Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction.

[103] Xiang Zhang and Yann Lecun. Text understanding from scratch. *arXiv: Learning*, 2015.

[104] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[105] Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. Aspect-augmented adversarial networks for domain adaptation. *Transactions of the Association for Computational Linguistics (TACL)*, 2017.