Ingredients for Happiness: Modeling constructs via semi-supervised content driven inductive transfer learning

by

Bakhtiyar Syed, Vijaysaradhi Indurthi, kulin Shah, Manish Gupta, Vasudeva Varma

in

THE AAAI-19 WORKSHOP ON AFFECTIVE CONTENT ANALYSIS
(*AFFCON-2019*)

Hilton Hawaiian Village, Honolulu, Hawaii, USA

Report No: IIIT/TR/2019/-1

Centre for Search and Information Extraction Lab
International Institute of Information Technology
Hyderabad - 500 032, INDIA
January 2019

Ingredients for Happiness: Modeling constructs via semi-supervised content driven inductive transfer learning

Bakhtiyar Syed*[1], Vijayasaradhi Indurthi*[1], Kulin Shah[1], Manish Gupta[1,2], and Vasudeva Varma[1]

[1] IIIT Hyderabad
{syed.b, vijaya.saradhi}@research.iiit.ac.in,
kulin.shah@students.iiit.ac.in, {manish.gupta, vv}@iiit.ac.in
[2] Microsoft
gmanish@microsoft.com

Abstract. Modeling affect via understanding the social constructs behind them is an important task in devising robust and accurate systems for socially relevant scenarios. In the CL-Aff Shared Task (part of Affective Content Analysis workshop @ AAAI 2019), the organizers released a dataset of 'happy' moments, called the HappyDB corpus. The task is to detect two social constructs: the agency (i.e., whether the author is in control of the happy moment) and the social characteristics (i.e., whether anyone else other than the author was also involved in the happy moment). We employ an inductive transfer learning technique where we utilize a pre-trained language model and fine-tune it on the target task for both the binary classification tasks. At first, we use a language model pre-trained on the huge WikiText-103 corpus. This step utilizes an AWD-LSTM with three hidden layers for training the language model. In the second step, we fine-tune the pre-trained language model on both the labeled and unlabeled instances from the HappyDB dataset. Finally, we train a classifier on top of the language model for each of the identification tasks. Our experiments using 10-fold cross validation on the corpus show that we achieve a high accuracy of ∼93% for detection of the social characteristic and ∼87% for agency of the author, showing significant gains over other baselines. We also show that using the unlabeled dataset for fine-tuning the language model in the second step improves our accuracy by 1-2% across detection of both the constructs.

Keywords: Happy Moments · Inductive transfer learning · Language model fine-tuning · Agency Prediction · Social Characteristic Detection

1 Introduction

In our quest to better model happy moments and characterize them, it is important to understand which entities were involved in the happy moments, and

* The authors contributed equally.

the psychology and behaviours which make people happy. Once the reasons and behaviours which trigger happiness are identified, techniques can be effectively developed to steer towards such behaviours which can increase people happiness levels. It is therefore useful to answer questions like (1) whether the author was in control of the happy moment (referred to as *agency* in this paper), and (2) whether multiple people contributed to the happy moment (referred to as social characteristic in this paper). The CL-AFF shared task at AffCon2019[3] focuses on answering these two research questions. Asai et al. [1] developed a database of 100K happy moments, HappyDB, using crowd sourcing and made it publicly available. We use this dataset to build models for answering the two questions.

Recently, there has been significant progress in the area of inductive transfer learning for natural language processing (NLP). Training deep learning models from scratch requires enormous amount of labeled data for achieving high accuracy. In recent times though, there have been advancements which give better performance on tasks like text classification from only a few labeled data instances [6].

In this work, we show that inductive transfer learning is greatly beneficial in identifying the agency and social characteristics of happy moments in the dataset. We also employ a variant wherein we utilize the 'unlabeled' happy moments and leverage it to increase the system performance. Our experiments using 10-fold cross validation on the corpus show that we achieve a high accuracy of $\sim$93% for detection of the social characteristic and $\sim$87% for agency of the author, showing significant gains over other baselines.

2 Problem Definition

We specifically attempt to solve Task 1 of the CL-Aff shared task, i.e., detecting the agency and social labels of the given happy moment. Formally, we model the task as follows.

Agency and Social characteristic detection: Given a happy moment H, we intend to learn the agency label $C1$ and the social label $C2$. $C1$ indicates whether the author is in control of the happy moment being described. $C2$, on the other hand, indicates whether anybody else other than the author, i.e., whether multiple entities are involved in the happy moment being described. We model both the tasks as binary classification problems. Thus, if author is in control of the happy moment, $C1$=1; otherwise, $C1$=0. Similarly, if anyone else other the author is involved in the happy moment, $C2$=1; otherwise, $C2$=0.

To solve these problems, we propose a semi-supervised inductive transfer learning approach. Our approach is inspired by the ULMFiT architecture [6] and AWD-LSTM [11] which we discuss in brief below.

[3] https://sites.google.com/view/affcon2019/cl-aff-shared-task

3 Preliminaries

In this section, we discuss the ULMFiT architecture and the AWD-LSTM model in brief.

3.1 The ULMFiT Architecture

Previous research has proposed multiple models for exploiting inductive transfer for Natural Language Processing (NLP) applications [5, 12]. In this work, we adapt a recently proposed architecture called ULMFiT (Universal Language Model Fine-tuning) for inductive transfer learning. The ULMFiT architecture proposed by Howard and Ruder [6] uses multiple heuristics for fine-tuning of language models (LMs) to avoid overfitting when training neural models on small labeled datasets. The ULMFiT architecture not just reduces the LM over-fitting but also prevents catastrophic forgetting of information which earlier models built on LMs were susceptible to. We adapt the ULMFiT model for our inductive transfer learning approach with a variant and show that inductive transfer learning is greatly beneficial for identifying agency and social characteristics of the happy moments in the given corpus. Besides exploiting just the labeled data, our variant also utilizes the unlabeled corpus for fine-tuning the language model which further improves the classification performance across both the constructs.

3.2 The AWD-LSTM Model

Our inductive transfer learning mechanism also makes use of the Averaged-SGD Weight-Dropped Long Short Term Memory (AWD-LSTM) networks [11]. The AWD-LSTM uses DropConnect and a variant of Average-SGD (NT-ASGD) along with several other well-known regularization strategies. We leverage the use of AWD-LSTMs as it has been shown to very effective in learning low-perplexity language models.

4 Approach: Inductive Transfer Learning

In this section, we describe the three phases of the proposed inductive transfer learning approach. Figure 1 illustrates the overall system architecture.

The proposed inductive transfer learning framework for identification of the 'agency' and 'social' characteristics makes use of the following three phases in order.

1. **General Domain Pre-training**: The first phase pre-trains the AWD-LSTM based language model on a huge text corpus. In our case, we use the pretrained language model trained on Wikitext-103 [11] dataset which consists of 103 Million unique words and 28,595 pre-processed Wikipedia articles. General domain pre-training helps the model learn basic characteristics of the language in question. It is essential that the LM be pre-trained on a huge corpus so that these general-domain characteristics are learned well.

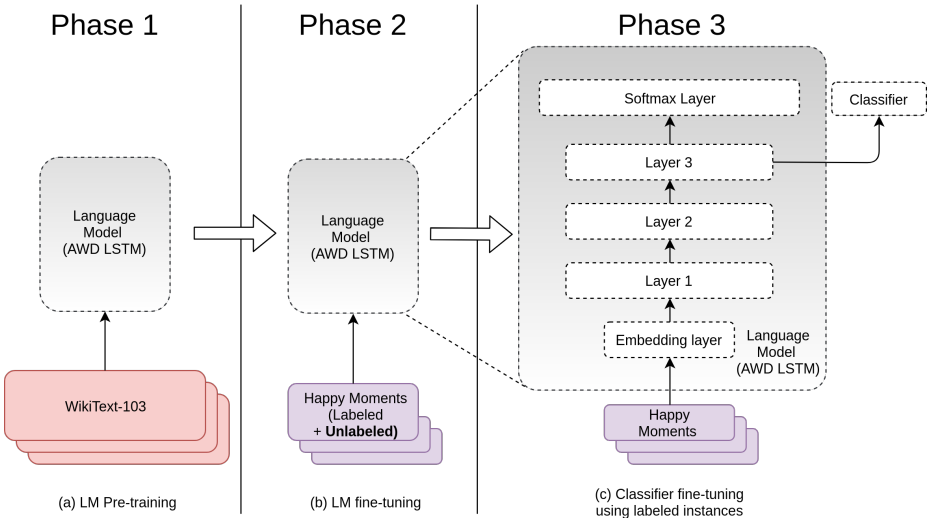

Fig. 1: Inductive Transfer Learning mechanism to identify agency and social characteristics. The same architecture is used for classifying *agent* and *social* characteristics separately.

2. **Language Model Fine-tuning for the Target Task**: For this step, after pre-training the language model with a huge corpus of the language texts, we fine-tune it using both the labeled as well as unlabeled part of the happy moments corpus. In this stage, we utilize task-specific data to fine-tune our language model in an unsupervised manner. As proposed in [6], our fine-tuning involves discriminative fine-tuning and slanted triangular learning rates to combat the catastrophic forgetting nature of language models as exhibited in previous works [5, 12]. In discriminative fine-tuning, instead of keeping the same learning rate for all the layers of the AWD-LSTM, a different learning rate is used for tuning the three different layers. The intuition is that since each of the layers represent a different kind of information [21], they must be fine-tuned to different extents. Also using the same learning rate is not the best way to enable the model to converge to a suitable region of the parameter space. Thus we adapt the slanted triangular learning rate [6] which first increases the learning rate and then linearly decays it as the number of training samples increases.

3. **Classifier Fine-tuning for the Target Task**: The weights that we obtain from the second phase are fine-tuned by extending the upstream architecture with two fully connected layers with softmax activation for the classification. In this phase, we adapt the *gradual unfreezing heuristic* [6] for our task. In gradual unfreezing (GU), all layers are not fine-tuned at the same time, instead the model is gradually unfrozen starting from the last layer, as it contains the least general knowledge [21]. The last layer is first unfrozen and fine-tuned for one epoch. Subsequently, the next frozen layer is unfrozen

and all unfrozen layers are fine-tuned. This is repeated until all layers are fine-tuned until convergence is reached.

5 Experiments

In this section, we describe the baselines and present comparisons between the baseline and our proposed approach.

5.1 Baselines

Word embedding is a technique in NLP which maps words of a language into dense vectors of real numbers in a continuous embedding space. Traditional NLP systems such as BoW (Bag of Words) and TF-IDF (Term Frequency-Inverse Document Frequency) are mainly syntactic representations and cannot capture the semantic relationships between words. Word embedding techniques have been gaining popularity in a range of NLP tasks like Sentiment analysis [10, 20], Named Entity Recognition [8, 18], Question Answering [15], etc.

As baselines, we use word embeddings and demographics features of the author of the happy moment like age, country, gender, marital status, parenthood, happiness duration. We train multiple classifiers using these set of features. Specifically, for the baselines, we use the following pre-trained word and sentence embedding models: GloVe, Concatenated Power Mean, Google Universal Embedding, fastText, Lexical Vectors and InferSent embeddings.

For word based embeddings, the embedding of the sentence is computed by tokenizing the sentence into words and computing the average of all the embeddings of the words of the sentence. We formulate the problem of identifying the social and agency attributes as text classification tasks. Hence, we use multiple supervised learning algorithms like Logistic Regression (LR), Support Vector Machines (SVM), Random Forests (RF), Neural Networks (with two hidden layers), and boosting (XGB) to train the models.

In the following, we describe the word/sentence embeddings which we use as baselines.

(1) fastText [2]: It is a skipgram based word embedding method, where each word is represented as a bag of character n-grams. A vector representation is associated to each character n-gram; words being represented as the sum of these representations.

(2) GloVe [13] is an unsupervised learning algorithm for distributed word representation. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. We use the standard 300 dimensional GloVe embeddings (GloVe1) trained on 840B word tokens. As another baseline, we also use 200 dimensional GloVe embeddings trained on a Twitter corpus (GloVe2) containing 27B word tokens.

(3) InferSent [4] is another set of embeddings trained by Facebook. InferSent is trained using the task of language inference. Given two sentences the model

is trained to infer whether they are a contradiction, a neutral pairing, or an entailment. The output is an embedding of 4096 dimensions.

(4) Concatenated Power Mean Word Embedding [16] generalizes the concept of average word embeddings to power mean word embeddings. The concatenation of different types of power mean word embeddings considerably closes the gap to state-of-the-art methods mono-lingually and substantially outperforms many complex techniques cross-lingually.

(5) Lexical Vectors [17] is another word embedding similar to fastText with slightly modified objective. Fast Text [2] is another word embedding model which incorporates character n-grams into the skipgram model of Word2Vec and considers the subword information.

(6) The Universal Sentence Encoder [3] encodes text into high dimensional vectors. The model is trained and optimized for greater-than-word length text, such as sentences, phrases or short paragraphs. It is trained on a variety of data sources and a variety of tasks with the aim of dynamically accommodating a wide variety of natural language understanding tasks. The input is variable length English text and the output is a 512 dimensional vector.

For each of the embeddings in the above list, we train models using different supervised learning algorithms. We use the scikit-learn implementations of these algorithms with the standard default parameters without any hyper-parameter tuning.

Model	**LR**			**RF**			**NN**			**SVM**			**XGB**		
	Acc.	F1	AUC	Acc.	F1	AUC	Acc.	F1	AUC	Acc.	F1	AUC	Acc.	F1	AUC
Universal	84.98	84.77	79.29	82.35	80.49	70.12	85.30	84.75	79.17	83.83	83.55	77.92	83.12	82.21	73.91
GloVe1	75.37	72.41	60.29	75.17	70.37	57.28	75.17	71.91	59.52	85.30	71.84	59.43	85.26	70.81	57.58
GloVe2	82.80	82.11	74.42	81.55	79.3	68.3	81.87	81.92	73.8	82.63	81.64	73.75	82.77	81.68	72.85
fastText	74.57	71.07	58.52	74.90	69.67	56.43	75.44	71.37	58.78	74.50	64.58	51.52	76.03	70.77	57.51
InferSent	80.00	78.72	73.54	85.20	82.66	73.44	83.27	80.90	74.91	79.35	78.27	72.9	85.84	84.51	77.09
LexVec	82.28	81.57	73.78	80.71	77.65	65.90	82.27	80.94	72.54	81.95	81.16	73.2	81.17	79.54	69.45
Concatenated p-means	76.36	76.61	70.63	83.27	82.20	73.13	80.77	80.95	75.66	76.44	76.74	70.89	83.81	82.98	74.96

Table 1: 10-fold cross validation Accuracy, F1 and AUC scores for *Agency* labels for the baseline models (embeddings+demography features)

Model	**LR**			**RF**			**NN**			**SVM**			**XGB**		
	Acc.	F1	AUC	Acc.	F1	AUC	Acc.	F1	AUC	Acc.	F1	AUC	Acc.	F1	AUC
Universal	91.51	91.51	91.51	90.93	91.21	91.3	92.04	91.95	91.99	91.70	90.86	90.88	90.82	90.83	90.86
GloVe1	81.05	81.01	81.48	78.89	79.14	79.12	81.13	80.97	81.48	80.61	80.40	81.01	81.86	81.72	82.59
GloVe2	88.80	88.81	88.87	87.98	87.89	87.96	89.30	88.89	88.94	88.92	88.62	88.72	88.15	88.16	88.25
fastText	79.49	79.39	80.11	78.15	78.18	78.13	79.83	79.79	80.56	79.27	79.26	80.18	80.85	80.68	81.64
InferSent	86.40	86.76	86.71	87.54	88.63	88.70	88.45	88.97	89.04	85.53	85.84	85.77	89.64	89.93	90.04
LexVec	88.88	88.89	88.94	87.58	87.69	87.91	88.70	88.78	88.90	88.55	88.69	88.78	88.42	88.43	88.55
Concatenated p means	85.11	85.15	85.06	88.57	88.25	88.29	89.66	89.06	89.07	84.49	84.42	84.35	89.78	89.79	89.91

Table 2: 10-fold cross validation Accuracy, F1 and AUC scores for *Social* labels for the baseline models (embeddings+demography features)

5.2 Hyper-parameter Settings

As suggested in [6], we use the AWD-LSTM language model with three layers, 1150 hidden activations per layer and an embedding size of 400. The hidden layer of the classifier is of size 50. A batch size of 30 is used to train the model. The LM and classifier fine-tuning is done with a base learning rate of 0.004 and 0.01 respectively. We built separate models for the 'agency' and 'social' classification tasks.

5.3 Results and Analysis

Tables 1 and 2 show the accuracy of the models trained on different word embeddings across various machine learning algorithms for the Agency and Social label prediction tasks respectively. Table 3 shows the performance of the inductive transfer model utilizing the unlabeled corpus for LM fine-tuning as compared with not utilizing the unlabeled corpus. We see that making use of the unlabeled corpus has its advantages as it gives the highest accuracy across both *agency* and *social* detection, beating other baselines strongly and outperforming the inductive transfer model used when not making use of the unlabeled corpus.

Table 3: 10-fold cross-validation Performance using Inductive Transfer Learning for both Agency and Social Characteristics Detection Tasks

Unlabeled data used in LM fine-tuning?	Agency			Social		
	Accuracy	F-1	ROC-AUC	Accuracy	F-1	ROC-AUC
No	85.97	85.45	78.93	92.1	92.11	92.23
Yes	86.45	86.16	80.18	92.7	92.69	92.71

6 Conclusions

In this work, we showed that the idea of using inductive transfer learning by fine-tuning language models helps in giving robust performance across detection of agency and social characteristics. We also showed that the use of unlabeled data for LM fine-tuning in our second stage helped in improving performance across 10-fold cross validation evaluation measures for both the tasks.

We plan to perform the given text classification using other pre-trained embeddings like ELMo (Embeddings from Language Models) [14], Skip-Thought Vectors [7], Quick-Thoughts [9] and Multi-task learning based sentence representations [19], and investigate if use of those embeddings can improve the classification accuracy. We would also like to experiment with other semi-supervised techniques to improve the classification accuracy.

References

1. Asai, A., Evensen, S., Golshan, B., Halevy, A., Li, V., Lopatenko, A., Stepanov, D., Suhara, Y., Tan, W.C., Xu, Y.: Happydb: A corpus of 100,000 crowdsourced happy moments. In: Proceedings of LREC 2018. European Language Resources Association (ELRA), Miyazaki, Japan (May 2018)
2. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606 (2016)
3. Cer, D., Yang, Y., Kong, S.y., Hua, N., Limtiaco, N., John, R.S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., et al.: Universal sentence encoder. arXiv preprint arXiv:1803.11175 (2018)
4. Conneau, A., Kiela, D., Schwenk, H., Barrault, L., Bordes, A.: Supervised learning of universal sentence representations from natural language inference data. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. pp. 670–680. Association for Computational Linguistics, Copenhagen, Denmark (September 2017), https://www.aclweb.org/anthology/D17-1070
5. Dai, A.M., Le, Q.V.: Semi-supervised sequence learning. In: Advances in neural information processing systems. pp. 3079–3087 (2015)
6. Howard, J., Ruder, S.: Universal language model fine-tuning for text classification. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 328–339 (2018)
7. Kiros, R., Zhu, Y., Salakhutdinov, R.R., Zemel, R., Urtasun, R., Torralba, A., Fidler, S.: Skip-thought vectors. In: Advances in neural information processing systems. pp. 3294–3302 (2015)
8. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360 (2016)
9. Logeswaran, L., Lee, H.: An efficient framework for learning sentence representations. arXiv preprint arXiv:1803.02893 (2018)
10. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1. pp. 142–150. Association for Computational Linguistics (2011)
11. Merity, S., Keskar, N.S., Socher, R.: Regularizing and optimizing lstm language models. arXiv preprint arXiv:1708.02182 (2017)
12. Mou, L., Meng, Z., Yan, R., Li, G., Xu, Y., Zhang, L., Jin, Z.: How transferable are neural networks in nlp applications? arXiv preprint arXiv:1603.06111 (2016)
13. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
14. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. arXiv preprint arXiv:1802.05365 (2018)
15. Ren, M., Kiros, R., Zemel, R.: Exploring models and data for image question answering. In: Advances in neural information processing systems. pp. 2953–2961 (2015)
16. Rücklé, A., Eger, S., Peyrard, M., Gurevych, I.: Concatenated p-mean word embeddings as universal cross-lingual sentence representations. arXiv preprint arXiv:1803.01400 (2018)
17. Salle, A., Villavicencio, A.: Incorporating subword information into matrix factorization word embeddings. arXiv preprint arXiv:1805.03710 (2018)

18. Santos, C.N.d., Guimaraes, V.: Boosting named entity recognition with neural character embeddings. arXiv preprint arXiv:1505.05008 (2015)
19. Subramanian, S., Trischler, A., Bengio, Y., Pal, C.J.: Learning general purpose distributed sentence representations via large scale multi-task learning. arXiv preprint arXiv:1804.00079 (2018)
20. Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., Qin, B.: Learning sentiment-specific word embedding for twitter sentiment classification. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 1555–1565 (2014)
21. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Advances in neural information processing systems. pp. 3320–3328 (2014)