Mitigating Temporal-Drift: A Simple Approach to Keep NER Models Crisp

Shuguang Chen †, **Leonardo Neves** ‡ and **Thamar Solorio** †
University of Houston †
Snap Research ‡
schen52@uh.edu, lneves@snap.com and tsolorio@uh.edu

Abstract

Performance of neural models for named entity recognition degrades over time, becoming stale. This degradation is due to temporal drift, the change in our target variables' statistical properties over time. This issue is especially problematic for social media data, where topics change rapidly. In order to mitigate the problem, data annotation and retraining of models is common. Despite its usefulness, this process is expensive and time-consuming, which motivates new research on efficient model updating. In this paper, we propose an intuitive approach to measure the potential trendiness of tweets and use this metric to select the most informative instances to use for training. We conduct experiments on three state-of-the-art models on the Temporal Twitter Dataset. Our approach shows larger increases in prediction accuracy with less training data than the alternatives, making it an attractive, practical solution. [1]

1 Introduction

Prediction performances of live machine learning systems degrade over time due to changes in the statistical properties of the data used for training them. This degradation, also known as temporal drift, happens in different ML tasks, including named entity recognition (NER). Due to the nature of the task, authors also call this language drift (Fromreide et al., 2014; Derczynski et al., 2015). Temporal drift effects are amplified in social media. Due to the ecosystem's very nature, topics reflect events and interests of a diverse user base and are continuously and rapidly evolving. To study the impact of language drift, we focus our analysis on the case of NER on Twitter data. Emerging and Trending topics are an essential part of Twitter. They change quite rapidly, reflecting diverse topics and world events of interest. Entities are a significant component of these changes, generating a diverse set of NE tokens. These ever-evolving topics pose a challenge as new entities frequently arise. The new entities are especially problematic as they might not exist in our previous vocabulary or can potentially transform the meaning of a previously observed term. Figure 1 shows tweets that include the emerging topic 'US'. After the release of the film, the topic 'US' became trending and aroused wide discussion. To mitigate the impact of temporal drift, we investigate how to effectively and efficiently adapt an already trained NER model to sustain prediction performance over time. We propose an intuitive approach to measure the trendiness of tweets and use this metric to select the most informative instances for retraining. We show that labeling instances based on this approach can yield better downstream performance than randomly sampling tweets for annotation.

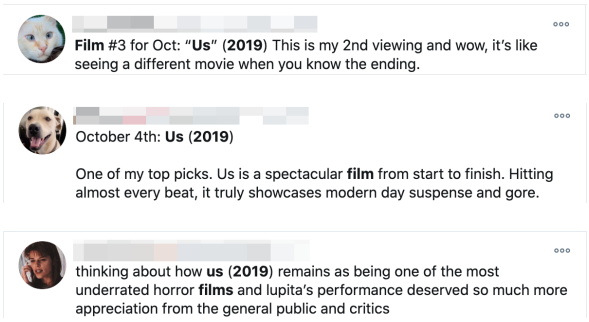


Figure 1: Examples of tweets that include the emerging topic '**US**', a horror movie released in 2019

Note that topics such as semantic shift (Hamilton et al., 2016; Rosenfeld and Erk, 2018) and active learning (Sinha et al., 2019; Kirsch et al., 2019) are related to the work we present here. In semantic shift, the core problem is how to trace temporal changes in lexical semantics, including linguistic drifts and cultural shifts. Unlike this task, our goal is to leverage the emergence of trends to guide an

[1] We release the code at https://github.com/RiTUAL-UH/trending_NER.

Proceedings of the Ninth International Workshop on Natural Language Processing for Social Media, pages 163–169
Online Workshop, June 10, 2021.

already trained model.

In active learning, researchers have focused on incremental annotation of instances by selecting the most informative ones. The goal is to achieve better results than random sampling. Multiple approaches exist to measure the informativeness of data points, but all of them are domain agnostic (Sinha et al., 2019; Kirsch et al., 2019). Our proposed solution is more straightforward than using uncertainty in ensembles or adversarial networks. However, it effectively increases model performance, and, similar to active learning approaches, it is more efficient than random sampling.

To summarize, we make the following contributions:

1. We propose an approach to measure the potential trendiness of tweets for selecting the most informative training samples.
2. We conduct extensive experiments and demonstrate the effectiveness of our approach for retraining a NER model.

2 Emerging Trend Detection

We want to exploit social media's inherent characteristics (Benhardus and Kalita, 2013; Mathioudakis and Koudas, 2010), with a focus on Twitter, to update model parameters efficiently. We assume that named entities associated with posts that are likely to become trends will be more informative and result in larger performance gains. Our emerging trend detection strategy is based on contrasting frequency of words in older data (training data) against frequency in newly collected data (recent data). More specifically, we formulate this task as detection of trending n-grams. We compute the trend scores for each n-gram, n, as follows:

$$score(n) = \frac{f_{n,R} - f_{n,P}}{f_{n,P} + k}$$

where $f_{n,R}$ and $f_{n,P}$ are the frequencies of n-gram n in the recent and past datasets, respectively. In practical applications, $f_{n,R}$ can refer to the frequency in newly collected data, while $f_{n,P}$ can refer to the frequency in older data. k is a normalization term used to mitigate the frequency of the highly-frequent n-grams in the recent datasets. When computing trend scores, we filter out stop words as they are usually the most common words but contain the least information. After we compute trend scores for all n-grams in newly collected data, we assign trend scores to the instances by summing over the scores of each n-gram in that instance (tweet). We then use the score to rank instances for labeling and updating the NER model. Our approach is flexible as it can be used in combination with any NER model architecture.

3 Experiments

We empirically study the impact of retraining NER models on trending data in two different scenarios. In **Scenario 1**, we retrain the model in an incremental manner with N instances from a newer batch of data in the following year at every iteration. In **Scenario 2**, we retrain the model incrementally as well, but the pool of data we used to select instances includes all years available in the training partitions. In both cases, instances are selected based on their trend scores.

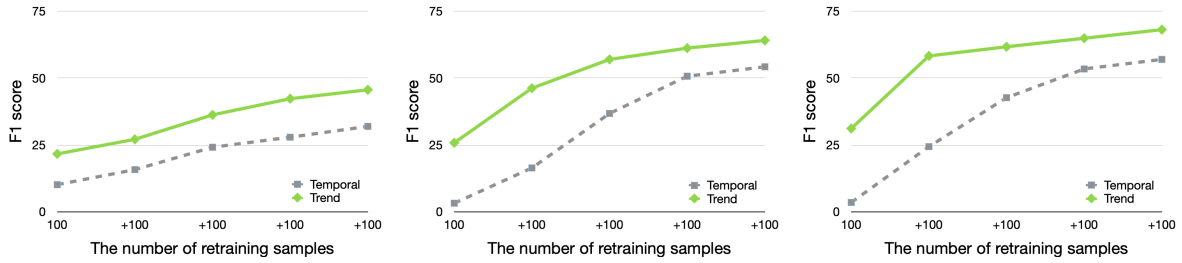
We use the Temporal Twitter Dataset from Rijhwani and Preotiuc-Pietro (2020) for all experiments. This dataset is temporally distributed and balanced with a variety of topics. It has 12K tweets collected from 2014 to 2019, with 2K samples from each year. In our experiments, the training set comprises of splits from 2014 to 2018. The validation set and test set have a random sample of 500 (25%) and 1,500 (75%) tweets from 2019, respectively.

3.1 Neural Architectures

As mentioned earlier, our approach is model agnostic. We validate this claim by experimenting with different NER neural architectures used in the prior art. The main difference between these models is the representation fed into a Conditional Random Field (CRF) (Lafferty et al., 2001) for prediction. The implementation and hyperparameters are described in Appendix A.

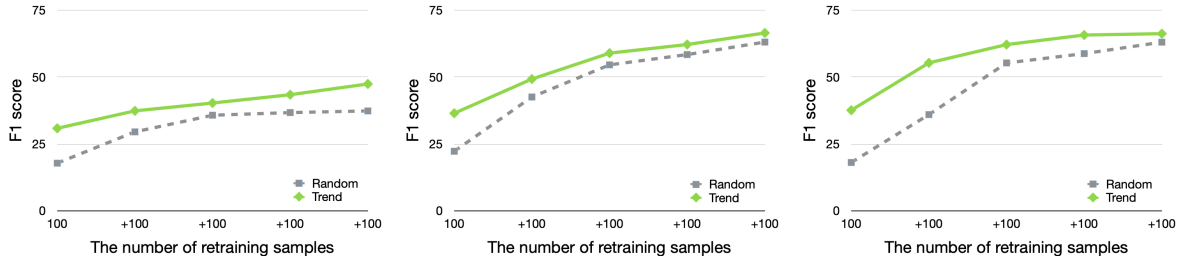
BiLSTM + CRF Following Ma and Hovy (2016), we use the GloVe (Pennington et al., 2014) word embeddings for word representations and Convolution Neural Networks (CNNs) for character representations. Then a bidirectional LSTM (Graves and Schmidhuber, 2005) takes both word representations and character representations as input and encodes sentences.

BERT + CRF BERT is a transformer-based model proposed by Devlin et al. (2019). It is pretrained using masked language modeling and next sentence prediction objectives on the corpora from the general domain. BERT takes subwords as input



(a) BiLSTM + CRF (b) BERT + CRF (c) BERTweet + CRF

Figure 2: **Data can only be accessed year by year -** Each step represents a year from 2014 to 2018. At each step, we add instances from its respective year to the training set. For Temporal, we randomly select instances from that given year. For Trend, we rank instances based on their trending score. We experiment with 50 (Appendix B), 100 and 200 (Appendix B) instances per step to show the impact of training size.



(a) BiLSTM + CRF (b) BERT + CRF (c) BERTweet + CRF

Figure 3: **Data from all years is available -** At each step, we add new instances to our training set. For Random, we randomly select instances from the available data. For Trend, we rank all available instances from most trending to less trending based on their trending scores. We then use this ranking to select the instances. At each step, we choose the instances with the highest trending scores that have not yet been added to the training set. We experiment with 50 (Appendix B), 100 and 200 (Appendix B) instances per step to show the impact of training size.

and generates contextualized word representations for each sentence.

BERTweet + CRF Similar to BERT, BERTweet (Nguyen et al., 2020) is a large-scale language model with the same configuration as BERT. It is pre-trained on the corpora from the social media domain and achieves state-of-the-art results on many downstream Twitter NER tasks.

3.2 Results

We empirically examine the performance of models under the influence of data evolution and temporal drift. We start with doing experiments on trending bi-grams and use the same amount of training samples at each step to eliminate the influence of training data size. Below we discuss the results of the two evaluation scenarios.

Scenario 1 In this scenario, we assume that the data can only be accessed chronologically by year. For each new batch of data selected based on the trendiness score (**trend**), we take the model as trained on the previous batch and retrain on the newest data. In other words, we consider the model from the previous iteration as a pre-trained model and fine-tune that model on the newest data. For comparison purposes, we run a **temporal** version, where the model is fine-tuned with newer data every time, but the instances are selected randomly for the corresponding year. Due to the randomness in this approach, we run each model five times, and then we report the average of the five runs as the final F1 score.

The results are shown in Figure 2. We observe that both temporal and trend F1 scores increase as we move temporally closer to the target data. However, in all cases, the trend-based models always reach a higher score.

Scenario 2 In this scenario, we assume the data can be accessed from all years at once. We merge the training data from all years and form a single

pool of data. However, we still fine-tune models at each iteration using the same number of new instances each time. For the trend models, we select instances based on their trend scores, regardless of the year, whereas for the random model, we select instances at random from the merged pool of data. Similar to what we did in scenario 1, we run each model 5 times and report the averaged results.

The results are shown in Figure 3. Similar to scenario 1, the F1 scores of the models trained on instances selected based on their trend scores are always higher than random sampling F1 scores. In addition, scenario 2, on average, works better than scenario 1, which is consistent with Rijhwani and Preotiuc-Pietro (2020). However, this setting requires the data available from all years from the very beginning. Compared to scenario 2, scenario 1 is far more realistic because it can be more easily applied in practice.

3.3 Analysis

Impact of training data size We ran additional experiments where we add different amounts of training data at each iteration (50 and 200). With less training data available, the benefits of selecting instances based on trend scores are amplified. Even if more data is available, using trend scores to select which instances to add always results in better performance than randomly choosing instances. Due to space limitations, the plots are in Appendix B figures 4, 5, 6 and 7.

Impact of pre-trained knowledge From figures 2 and 3, we observe that, in general, pre-trained models (BERT and BERTweet) tend to perform closer to that of the trend-based models. Apart from the well-documented advantages of contextualized representations, we believe that higher performance here is due to these models' pre-trained knowledge. We suspect that if we had the ability to control the data, and in particular, the year of the data used in pre-training, the results would be different, and we would observe a larger gap between pre-trained transformer models and the trend-based approach.

Entity-wise Model Performance We investigate whether our approach affects named entity types differently. To this end, we create random data and trending data. The random data is randomly selected, while the trending data is selected based on the trend scores. Each data has 1,000 samples. Table 1 shows the model performance on the random data, versus the trending data. We notice that all three models overall benefit from trend detection with an improvement from 2.70% 5.71% on F1 metric, indicating that the models can adequately learn the context of named entities.

Model	Random data			Trending data		
	P	R	F1	P	R	F1
BiLSTM + CRF	59.38	48.02	53.10	63.42	54.83	58.81
BERT + CRF	62.26	73.23	67.30	70.07	69.93	70.00
BERTweet + CRF	60.64	64.84	62.67	65.45	70.46	67.86

Table 1: Performance comparison on random data and trending data, including persc.

To better understand the high model performance on trending data, Table 2 shows the distribution of random and trending data. By selecting training samples based on our approach, the number of entities in the trending data is 77% more than the number of entities in the random data, including 92% more PER, 38% more LOC, and 91% more ORG. In the token level, there are more 108% entity tokens in the trending data than in the random data. The higher ratio of named entities in the trending data increases the diversity of each entity type, and therefore, decreases the test error.

Entity Type	Random data		Trending data	
	Entity-level	Token-level	Entity-level	Token-level
PER	225	340	432	755
LOC	178	226	245	362
ORG	281	379	537	848
Total	684	945	1,214	1,965

Table 2: The distribution of random data and trending data, including entity-level distribution (entity spans) and token-level distribution (entity tokens).

4 Related Work

Previous work has studied trend detection in online social media platforms such as Twitter and Facebook (Benhardus and Kalita, 2013; Mathioudakis and Koudas, 2010; Miot and Drigout, 2020). Benhardus and Kalita (2013) outlined the methodologies for using the data from online platforms and proposed criteria based on the frequency of words to identify trending topics in Twitter. Mathioudakis and Koudas (2010) presented a system to detect bursty keywords that suddenly appear in tweets at an unusually high rate. Recently, Miot and Drigout (2020) investigated the efficiency of deep neural networks to detect trends. However, these techniques are applied without taking named entities into consideration.

Towards emerging named entities, recent work has mainly focus on identification and classification of unusual and previously unseen named entities. Derczynski et al. (2015) investigated the effects of data drift and the evaluation of the NER models on temporally unseen data. Agarwal et al. (2018) studied the disambiguation of named entities with explicit consideration of temporal background. Rijhwani and Preotiuc-Pietro (2020) reported improvements on performance for overlapping named entities under the impact of temporal drift. Due to the limitation of resources and lack of annotated data from social media, these NER models tend to have lower performances on emerging named entities.

5 Conclusion

In this work, we propose a simple approach to update model parameters and prevent degradation performance from temporal drifts. Our approach is inspired by our observations of how Twitter data follows trends in topics that can change very quickly. Experimentally, we show that leveraging emerging trends can benefit the recognition of named entities and reduce performance degradation, especially in low-resource scenarios. Our proposal is model agnostic, and can potentially be adapted to other NLP tasks that target social media and face the same problems of data evolution and temporal drift.

Acknowledgements

This work was partially supported by the National Science Foundation (NSF) under grant #1910192. We would like to thank the members from the RiTUAL lab at the University of Houston for their invaluable feedback. We also thank the anonymous reviewers for their valuable suggestions.

References

Prabal Agarwal, Jannik Strötgen, Luciano del Corro, Johannes Hoffart, and Gerhard Weikum. 2018. diaNED: Time-aware named entity disambiguation for diachronic corpora. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 686–693, Melbourne, Australia. Association for Computational Linguistics.

James Benhardus and J. Kalita. 2013. Streaming trend detection in twitter. *Int. J. Web Based Communities*, 9:122–139.

Leon Derczynski, Isabelle Augenstein, and Kalina Bontcheva. 2015. USFD: Twitter NER with drift compensation and linked data. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 48–53, Beijing, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Hege Fromreide, Dirk Hovy, and Anders Søgaard. 2014. Crowdsourcing and annotating NER for Twitter #drift. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2544–2547, Reykjavik, Iceland. European Language Resources Association (ELRA).

A. Graves and J. Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm networks. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, 4:2047–2052 vol. 4.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Cultural shift or linguistic drift? comparing two computational measures of semantic change. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2116–2121, Austin, Texas. Association for Computational Linguistics.

Andreas Kirsch, Joost R. van Amersfoort, and Y. Gal. 2019. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. In *NeurIPS*.

J. Lafferty, A. McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

M. Mathioudakis and N. Koudas. 2010. Twittermonitor: trend detection over the twitter stream. In *SIGMOD Conference*.

Alexandre Miot and Gilles Drigout. 2020. An empirical study of neural networks for trend detection in time series. *SN Comput. Sci.*, 1:347.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, Online. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Shruti Rijhwani and Daniel Preotiuc-Pietro. 2020. Temporally-informed analysis of named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7605–7617, Online. Association for Computational Linguistics.

Alex Rosenfeld and Katrin Erk. 2018. Deep neural models of semantic shift. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 474–484, New Orleans, Louisiana. Association for Computational Linguistics.

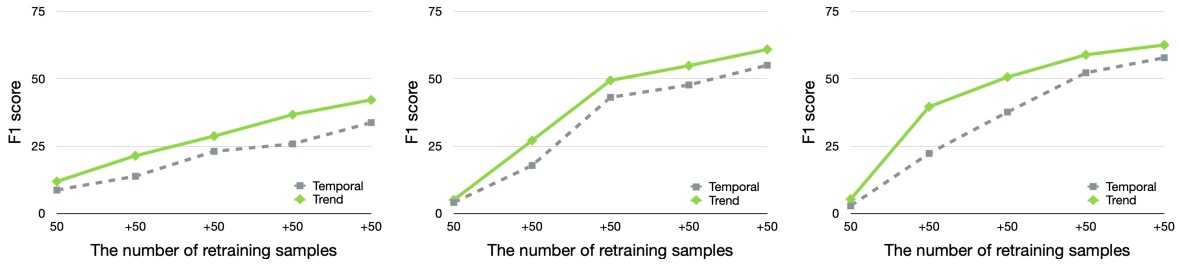
Samarth Sinha, S. Ebrahimi, and Trevor Darrell. 2019. Variational adversarial active learning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5971–5980.

A Details for Experimental Setup

For BiLSTM-CRF model, we use GloVe Twitter embeddings. The dimensions of character embeddings and word embeddings are 50 and 100 respectively. We then use 2-layer LSTM with 300 hidden units to encode sentences. The dropout rate is 0.5. During training, we use stochastic gradient descent (SGD) with learning rate 0.1, batch size 20, and momentum 0.9. The L2 regularization is set to 0.001. For BERT and BERTweet, we do fine-tuning using AdamW optimizer (Loshchilov and Hutter, 2017) with learning rate 5e-5, batch size 32, and weight decay 0.01. We also use a gradient clipping of 1.0 and the dropout rate is 0.1. In scoring function, k is set as 0.1 for sample selection.

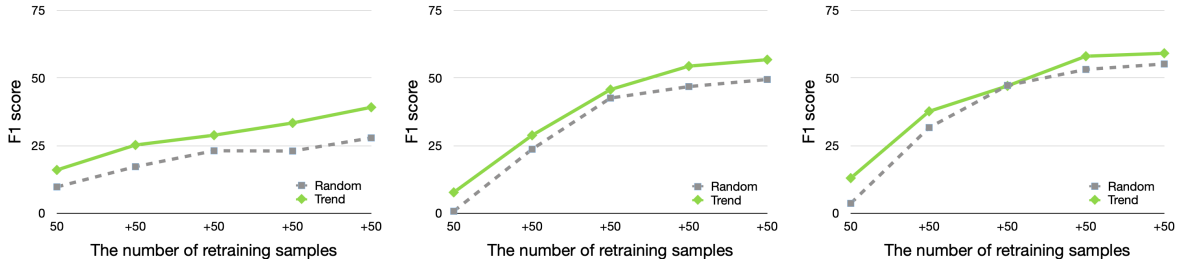
B Experiment with more data

In Figure 4 and Figure 5, we use 50 instances at each step. In Figure 6 and Figure 7, we use 200 instances at each step. We repeat our experiment with using different number of instances at each training step to study the impact of dataset size.



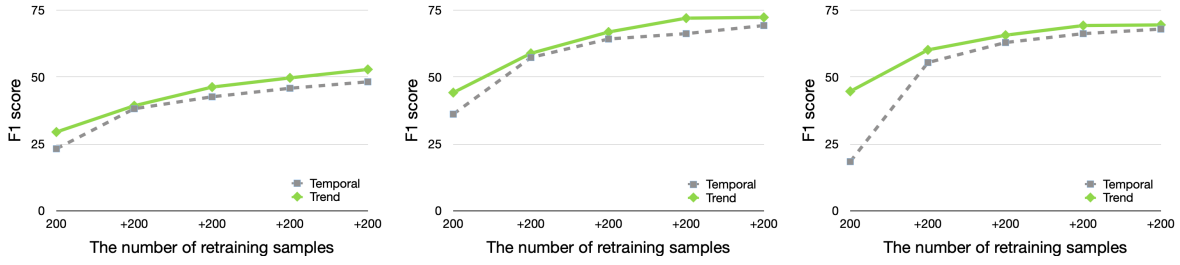
(a) BiLSTM + CRF (b) BERT + CRF (c) BERTweet + CRF

Figure 4: **Data can only be accessed year by year -** Each step represents a year from 2014 to 2018. At each step, we add instances from its respective year to the training set. For Temporal, we randomly select instances from that given year. For Trend, we rank instances based on their trending score. We experiment with 50 (Appendix B), 100 and 200 (Appendix B) instances per step to show the impact of training size.



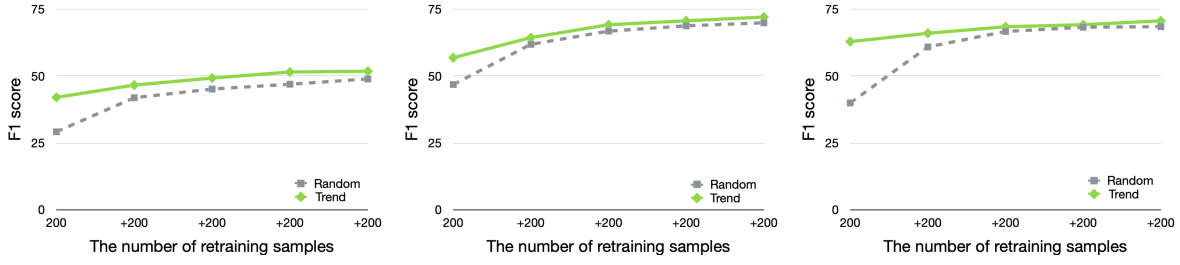
(a) BiLSTM + CRF (b) BERT + CRF (c) BERTweet + CRF

Figure 5: **Data from all years is available -** At each step, we add new instances to our training set. For Random, we randomly select instances from the available data. For Trend, we rank all available instances from most trending to less trending based on their trending scores. We then use this ranking to select the instances. At each step, we choose the instances with the highest trending scores that have not yet been added to the training set. We experiment with 50 (Appendix B), 100 and 200 (Appendix B) instances per step to show the impact of training size.



(a) BiLSTM + CRF (b) BERT + CRF (c) BERTweet + CRF

Figure 6: **Data can only be accessed year by year -** Each step represents a year from 2014 to 2018. At each step, we add instances from its respective year to the training set. For Temporal, we randomly select instances from that given year. For Trend, we rank instances based on their trending score. We experiment with 50 (Appendix B), 100 and 200 (Appendix B) instances per step to show the impact of training size.



(a) BiLSTM + CRF (b) BERT + CRF (c) BERTweet + CRF

Figure 7: **Data from all years is available -** At each step, we add new instances to our training set. For Random, we randomly select instances from the available data. For Trend, we rank all available instances from most trending to less trending based on their trending scores. We then use this ranking to select the instances. At each step, we choose the instances with the highest trending scores that have not yet been added to the training set. We experiment with 50 (Appendix B), 100 and 200 (Appendix B) instances per step to show the impact of training size.