


TSPA: Efficient Target-Stance Detection on Twitter

Evan M. Williams
School of Computer Science
Carnegie Mellon University
 Pittsburgh, PA
 emwillia@andrew.cmu.edu 

Kathleen M. Carley
School of Computer Science
Carnegie Mellon University
 Pittsburgh, PA
 kathleen.carley@cs.cmu.edu

Abstract—Target-stance detection on large-scale datasets is a core component of many of the most common stance detection applications. However, despite progress in recent years, stance detection research primarily occurs at the document-level on small-scale data. We propose a highly efficient Twitter Stance Propagation Algorithm (TSPA) for detecting user-level stance on Twitter that leverages the social networks of Twitter users and runs in near-linear time. We find TSPA achieves SoTA accuracy against BERT, homogenous Graph Attention Networks (GAT), and heterogenous GAT baselines. Additionally, TSPA’s wall-clock time was 10x faster than our best baseline on a GPU and over 100x faster than our best baseline on a CPU.

Index Terms—Stance, Twitter, Networks, GNNs, Label Propagation

I. INTRODUCTION

Target Stance Identification on Twitter, identifying the stance of accounts towards a specific target or topic, is a core component of many NLP tasks including opinion mining, hate-speech detection, and rumor detection. Until recently, nearly all target-stance classification benchmarks have been labeled at the document level. While this is useful in some domains, document-level stance alone is often insufficient given the adversarial nature of many of the most common target-stance classification applications. In market-trend analysis, opinion mining, content moderation, users often have motives to manipulate the information environment, and documents are not evenly created across users. In social media market-trend analysis, a single spamming user or bot network could sway market assessments. In hate-speech detection, users frequently develop hateful coded language to avoid automated censors, making language models alone potentially insufficient [1]. In information operations, many accounts are created, often by state-backed actors, to spread false, biased, or misleading information that could appear to be legitimate opinions held by citizens to pollsters or politicians. In each of these cases, it is important to understand the broader contexts of accounts to make informed decisions.

While substantial progress has been made in stance detection in recent years, we focus on two open problems. Firstly, target-stance detection on Twitter has been treated primarily as a text-classification task. However, practitioners are often interested in stance at a user-level. We find that the social networks of Twitter users contain very strong user-level stance signals that, when leveraged, can greatly improve

performance. For example, retweets can often be read as endorsements or agreements, whereas mentions or replies may be used to express agreement, but also many be used to express disagreement, nuance, or may be used change topics altogether. Secondly, SoTA deep learning approaches generally require GPU access and even with GPUs, models can take a long time to train. For practitioners operating in dynamic environments, time and resources can often be scarce.

We propose the Twitter Stance Propagation Algorithm (TSPA), a modified label propagation algorithm that leverages different interaction types on Twitter to propagate stance through weakly-labeled Twitter networks. We find, surprisingly, that simple label propagation through users’ social networks can outperform SoTA text and user classification approaches by a wide margin. In experiments on large-scale Twitter stance datasets, we find that TSPA can outperform both Graph Neural Networks and BERT-based classifiers on large-scale stance data. Additionally, our implementation runs in near-linear time and does not use a GPU. In our experiments, we find TSPA’s wall clock time was 10x faster than our top neural baseline model, a heterogenous graph attention network (GAT). When the GAT was instead run on a CPU, TSPA’s wall-clock time was 113x faster¹. We submit this work in the hope of making stance detection more accessible to organizations in dynamic environments or without GPU access.

II. RELATED WORK

User-level target-stance classification research is typically done on small-scale datasets. User-level stance papers most often rely on the data from Semeval-2016 Task 6 [2], a Twitter user-level stance detection task, and on PHEME, a rumor verification dataset [3]. While these are well-constructed datasets with reliable labels, practitioners often need to work with millions of documents at a time and need to consider scalability of algorithms.

Several researchers have explored how social networks and user features can be leveraged in target stance detection. Retweet networks have been associated with stance for years and have been used to build networks of users with similar stances [4], [5]. Other researchers have leveraged

¹CPU experiments were run on an Intel Xeon W-1350 processor and GPU experiments were run on a NVIDIA GeForce RTX 3080

social networks alongside features in stance detection task. [6] use a random walk graph to transform user-level bag-of-words or interactions graph into a feature similarity space for stance detection. In later work, [7] introduced an unsupervised approach to user stance detection on Twitter that created low-dimensional feature vectors for users and then clustered those vectors. In very recent work, researchers have begun leveraging Graph Neural Network models for stance detection [8], [9]. For more in-depth overviews of stance detection, [10] provides an excellent survey overview, [11] provides an overview of stance approaches used on social media data, and [12] provides a survey on stance detection in misinformation and disinformation detection.

III. DATA

We use the South American regional protest datasets constructed in [13], which created weak labels for 550k users and 36 million Spanish-language Tweets. These data were captured in 2019 during the massive and widespread South American anti-austerity protests. The data labels capture public opinions towards the protests, and by extension, towards their governments in Bolivia, Ecuador, Chile, and Colombia. Binary pro-government or anti-government labels were weakly assigned to users who wrote tweets or had an account description containing tokens from a list of "stancetags," n-grams or hashtags associated uniquely with a certain stance. These stancetags were collected from the Twitter accounts of partisan politicians from each of these countries. The full methodology details can be found in [13]. The Chile dataset was larger than the other three and would have required different hardware to preprocess. Consequently, we chose not include the Chilean data. Each country-level dataset contains retweets and "replies", where "replies" include Twitter replies, quotes, and mentions. Network statistics for each country can be found in Table I.

TABLE I
EDGE AND NODE STATISTICS

	Edges		Users	
	Retweets	Replies	Pro-	Anti-
Ecuador	2858300	543235	25567	51466
Bolivia	3874586	797214	58508	54347
Colombia	4570121	1101840	28204	79823

Practitioners working with stance or public opinion on social media often use stancetags to weakly label a subset of the data. This subset can then be used to build classifiers which can then be used to capture more data of the same stance. In this scenario, only a subset of the useful stancetags are initially known. Therefore, randomly sampling the South American Protest data would not reflect the initial state in practice, as the initial state would be constructed with stancetags not yet known by the practitioner.

To simulate a real-world use-case, we re-label users using a random subset of the initial stancetags used by the authors of the dataset. We aimed to create samples that captured

$20\% \pm 10\%$ of the data, as this is a plausible scenario for practitioners who are weakly labeling data. We observed that adding additional hashtags past this threshold had little impact on *TSPA*'s accuracy. We assign users stances based on whether or not they sent tweets containing the stancetags. In cases where authors sent two or more tags of different stance, authors were not assigned an initial stance. For the Ecuador data, we used 4 pro-government and 4 anti-government hashtags, resulting in 27% of the dataset being given labels. This initializes the model with 10,408 users labeled as "anti" and 10,231 labeled as "pro". Despite the imbalance in stance in the true distribution, *TSPA* appears to do well with relatively-balanced starting labels. For the Bolivian data, we used 5 pro-government and 5 anti-government stancetags, resulting in 17% of the dataset being given labels. For Colombia, we used 10 pro and anti hashtags, resulting in 11.6% coverage. The re-labeled data were then treated as the set of initial labels and evaluation was performed on the data that was not relabeled. In the Bolivian case, we evaluated each model on the remaining 83% of the data. For BERT and Graph neural network baselines, we create a 90-10 train-val split using the relabeled data, but again, evaluated on the not-relabeled test data.

IV. METHODS

Let G be a heterogeneous graph, $\mathcal{G} = (V, A, Z, \phi, Y)$ where V denotes nodes u_1, u_2, \dots, u_n . $A \in \mathbb{R}^{n \times n}$ is a row-normalized adjacency matrix. $Z \in \mathbb{R}^{n \times d}$ is a node feature matrix. To create features, we concatenate all Tweets for each user, lowercase the tweets, remove Spanish stopwords, and run truncated SVD on the resulting user-term frequency matrix. This generates a 100-dimensional feature vectors for each node. While we could use neural embedding approaches to create more complex feature vectors, we chose prioritize algorithmic efficiency. ϕ denotes a type of relation: in this work, either retweet network \mathcal{G}_{rt} or the full network containing retweets and responses \mathcal{G} where $\mathcal{G}_{rt} \subset \mathcal{G}$. Y is a one-hot encoded matrix of node labels. Our goal is assign stance labels to unlabeled users by learning a mapping from $\mathcal{G} \rightarrow L$.

A. Label Propagation Algorithm (RAK)

Label Propagation Algorithm (LPA) can refer to several different algorithms, e.g., ([14]–[16]). We build on the LPA approach proposed by [15], which, for disambiguation, has also been called RAK [17]. RAK assumes that labels of unlabeled nodes can be determined by the neighbors of that unlabeled node. In social networks, which have been shown to frequently exhibit homophily [18], [19], RAK can be highly effective [20]. The algorithm can be summarized as:

- 1) All nodes are randomly labeled
- 2) Nodes are shuffled into a random order
- 3) Each node u_i is assigned the label held by the majority of its neighbors. In the event of a tie, a label is randomly chosen.
- 4) If not converged, return to (2)

While this algorithm is not guaranteed to converge to a stationary solution, it executes in near-linear time, which is very beneficial when dealing with large real-world networks. [21] updated the algorithm to provably converge in semi-supervised settings. This, following the logic of [14], requires allowing nodes to "absorb" only fractions of neighbors weights at each time-step. However, this approach requires repeated iteration over the initial set of nodes, which increases time complexity. We chose to prioritise time complexity so that we could iterate only over unlabeled nodes at each iteration so that subsequent iterations would be much faster than the first. Despite the lack of provable convergence, we observed very little variance in accuracy in our final predictions, and found the approach to be useful empirically.

B. Twitter Stance Propagation Algorithm

Twitter Stance Propagation Algorithm (TSPA) modifies RAK to better leverage relations and feature attributes for users in Twitter. As with RAK, Y can be thought of as a one-hot encoded label matrix $Y^{(k)} = [y_1^{(k)}, y_2^{(k)}, \dots, y_n^{(k)}]^\top$ where row $y_i^{(k)\top}$ contains one-hot encoded labels of node u_i at iteration $k > 0$. When $k = 0$, $Y^{(0)}$ is a matrix of one-hot indicator vector labels for all nodes in V . Nodes without a label $V_{unlabeled}$ are assigned a 0 vector. However, TSPA modifies RAK in four key ways.

Firstly, TSPA is first applied to the retweet network \mathcal{G}_{rt} , and then, following convergence, the labels predicted from the retweet network become the new $Y^{(0)}$, and the algorithm is applied to the full network, \mathcal{G} . We do this to better leverage the strong stance signals present in the retweet network. Secondly, similar to [22], voting ties are not broken by random choice. Rather, if a node has an equal probability to be pro- or anti-government, it is skipped. This can result in a small subset of nodes not receiving labels. In practice, it can be nice to know where uncertainty lies, but for fairness of comparison, when calculating accuracy in experiments, we counted any unlabeled nodes as incorrectly classified. Thirdly, we weight neighbor votes by a feature similarity matrix $\mathcal{S} \in \mathbb{R}^{n \times n}$ so that neighbors with more similar features have more influential votes than neighboring nodes with more different features. \mathcal{S} in this work is the normalized pairwise cosine similarity matrix of the feature matrix Z . Finally, we only iterate over $V_{unlabeled}$, which shrinks at each iteration, resulting in each iteration being faster than the previous iteration. The labeling function for node i can be written as:

$$y_i = \operatorname{argmax}_l \sum_{j \in \mathcal{N}^{(l)}(i)} w_{ij} s_{ij} \quad (1)$$

where $\mathcal{N}^{(l)}(i)$ are the neighbors of i with label l . $w_{ij} s_{ij}$ are the similarity-weighted edge weights from $A \otimes \mathcal{S}$ where \otimes denotes the hadamard product. The algorithm stops when no new labels can be predicted. We include the pseudocode for TSPA in Algorithm 1. We also try running TSPA on only the retweet network, and then using the resulting labels to train a GNN. We denote Y following TSPA's convergence on \mathcal{G}_{rt} as $TSPA_{rt}$.

Algorithm 1 Twitter Stance Propagation Algorithm

input \mathcal{G}

weight the normalized adjacency matrix by the feature similarity matrix: $\mathcal{G}[A] \leftarrow \mathcal{G}[A] \otimes \mathcal{S}$

for each $g \in \mathcal{G}_\phi$ **do**

while $Y_{k-1} \neq Y_k$ **do**

shuffle $V_{unlabeled}$

for each $i \in V_{unlabeled}$ **do**

$\mathcal{N}^{(l)}(i)$ vote on label for i

if tie then

skip

else

$y_i = \operatorname{argmax}_l \sum_{j \in \mathcal{N}^{(l)}(i)} w_{ij} s_{ij}$

delete i from $V_{unlabeled}$

$\mathcal{G}[Y] \leftarrow Y$

C. Network Predictions

Formally, for node u , our goal is to create node embeddings $z_u \in \mathbb{R}^d$ that map node u to its corresponding one-hot-encoded stance label $y_u \in \mathbb{Z}^c$:

$$\mathcal{L} = \sum_{u \in V_{train}} -\log(\sigma(z_u, y_u)) \quad (2)$$

where σ is a softmax function. For graph-based baselines, we use homogeneous and heterogeneous Graph Attention networks, originally proposed by [23]. For homogeneous baselines, we use two Graph Attention Network (GAT) Layers with dropout, each with a single attention heads, followed by 4 linear layers with batchnorm, dropout, and relu nonlinearity. We use cross-entropy loss and log-softmax as our final activation function.

For heterogeneous GAT [24] baselines, we include whether relations are retweets or replies. We found that using our homogeneous architecture with heterogeneous GAT layers caused the model to overfit too quickly, so we removed 2 linear and batchnorm layers. For both our heterogeneous and homogeneous models, we use an Adam optimizer with weight decay, a learning rate of 1e-3 and a weight decay of 5e-4, a batch size of 512. We train all models over 100 epochs with early stopping based on validation loss and a patience of 15. Given the graph is too large to fit on a GPU, we used the inductive graph sampler proposed by [25] and implemented in pytorch geometric [26].

D. Text Predictions

For our text-classification baseline, we use BETO, the cased Spanish BERT model pretrained by [27]. We train with a batch size of 8 over 2 epochs. We save checkpoints every 500 steps and use validation loss to select the best checkpoint model. User-level text classification on Twitter data is a challenge because each user can have any number of tweets associated with their account. Additionally, we use stancetags to assign weak labels, so the model could simply learn to identify their presence, which not help classify any tweets in the test set. To

address these issues, we randomly sample 3 tweets for each user, with replacement, and concatenate them into a single representation for each user. We then remove all stancetags that were used for relabeling from the data. We could have chosen more advanced transformer-based approaches to embed users, but this would not advance our goal of increased computational efficiency.

V. RESULTS AND DISCUSSION

A. Main Results

TSPA outperformed all of our baseline models in experiments and did so with a fraction of the wall-clock time. In II, we compare our homogenous GAT, heterogenous GAT, and BERT-based baselines to several variants of TSPA. TSPA_{hom} is TSPA without relations ϕ , i.e., no distinction between the retweet network and the reply network. This clearly harms the model’s accuracy in all countries. TSPA_{uw} excludes weighting by feature similarity \mathcal{S} . Including weights had little effect on the overall accuracy of the models, but yielded several additional accuracy points when run only on the reply networks. Finally, we allowed TSPA to converge on the retweet network and then used those labels to seed a homogenous GAT network. This resulted in the best accuracy for Bolivia, but was still inferior to TSPA in Ecuador and Colombia. In Ecuador and Colombia, TSPA_{rt} followed by a homogenous GAT still resulted in overfitting on the training data.

TABLE II
ACCURACY OF TSPA VARIANTS AND BASELINES

	Bolivia	Ecuador	Colombia
Chile			
hom-GAT	0.833	0.890	0.858
het-GAT	0.929	0.900	0.847
BERT	0.623	0.651	0.765
TSPA _{hom}	0.925	0.944	0.956
TSPA _{uw}	0.949	0.954	0.963
TSPA	0.950	0.953	0.963
TSPA _{rt} + hom-GAT	0.960	0.914	0.819

We compare the wall-clock time of TSPA compared to our benchmarks. We excluded the data import and SVD calculations from the time measurements, as these are identical for the GAT and TSPA baselines and were precomputed. However, the time taken to construct the graph data objects are included in run-time measurements. As one of our goals was to create a fast algorithm that is not reliant on GPU, we chose to evaluate run time with and without GPU access. We found that when all baselines were trained on only a CPU, TSPA was over 100x faster than our best GAT baseline. When our GAT models were evaluated with a GPU, TSPA was still 1.97 times faster than the homogenous GAT and 10.64 times faster than the heterogenous GAT. BERT on a GPU was over 253x slower than TSPA and far less accurate, so we saw no reason to assess BERT run time without a GPU. We present the run times for each model on the Bolivian data in Table III. TSPA ratio is simply the wall-clock time of the algorithm in seconds divided by the wall-clock time of TSPA in seconds.

TABLE III
WALL-CLOCK TIME IN SECONDS FOR TSPA AND BASELINES

	Run Time	TSPA Ratio	Accuracy
TSPA (cpu)	97.39	1.00	0.950
hom-GAT (gpu)	191.77	1.97	0.833
hom-GAT (cpu)	9754.73	100.17	0.833
het-GAT (gpu)	798.88	10.64	0.929
het-GAT (cpu)	11081.29	113.79	0.929
BERT (gpu)	48706	253.98	0.623

VI. LIMITATIONS AND FUTURE WORK

With truncated SVD, we lose local and contextual signals that are likely important in determining stance. In future work, we plan to explore how transformers can be used in conjunction with GNNs to create representations of users that better incorporates context and other account features.

Additionally, the dataset we use only considers pro- and anti- government labels. However, it’s still possible users have nuanced views, change views, or have no views at all. Hashtag Hijack, when one group repurposes another group’s hashtag, is a big issue in weak labeling, and may result in some users being inaccurately labeled. Additionally, We do not evaluate the impact of imbalanced stance tags. Finally, given the propagation-based nature of the algorithm, it may be less accurate for smaller components if the graph is not fully connected.

VII. CONCLUSIONS

We proposed Twitter Stance Propagation Algorithm (TSPA), an algorithm for efficiently propagating stance labels by leveraging heterogeneous social network relations on Twitter. TSPA outperformed homogenous GAT, heterogenous GAT, and BERT baselines. TSPA’s wall-time was 10x faster than the best benchmark, a heterogenous GAT, when the heterogenous GAT ran on a GPU and TSPA’s wall-time was over 100x faster when the heterogenous GAT was run on a CPU. We hope that work helps in making stance detection more accessible to organizations in dynamic environments or without GPU access.

VIII. ACKNOWLEDGEMENTS

The research for this paper was supported in part by the ARMY Scalable Technologies for Social Cybersecurity, Office of Naval Research, MURI: Persuasion, Identity, Morality in Social-Cyber Environments, and Office of Naval Research, MURI: Near Real Time Assessment of Emergent Complex Systems of Confederates under grants W911NF20D0002, N000142112749, and N000141712675. It was also supported by the center for Informed Democracy and Social-cybersecurity (IDeaS) and the center for Computational Analysis of Social and Organizational Systems (CASOS) at Carnegie Mellon University. The views and conclusions are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ARMY, the ONR, or the US Government.

REFERENCES

- [1] N. Sonnad, "Alt-right trolls are using these code words for racial slurs online," *Quartz*. Available at: <https://qz.com/798305/alt-right-trolls-are-using-google-yahooskittles-and-skypes-as-code-words-for-racial-slurs-on-twitter/>, 2016.
- [2] S. Mohammad, S. Kiritchenko, P. Sobhani, X. Zhu, and C. Cherry, "SemEval-2016 task 6: Detecting stance in tweets," in *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, 2016, pp. 31–41.
- [3] E. Kochkina, M. Liakata, and A. Zubiaga, "All-in-one: Multi-task learning for rumour verification," *arXiv preprint arXiv:1806.03713*, 2018.
- [4] J. Borge-Holthoefer, W. Magdy, K. Darwish, and I. Weber, "Content and network dynamics behind egyptian political polarization on twitter," in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, 2015, pp. 700–711.
- [5] I. Weber, V. R. K. Garimella, and A. Batayneh, "Secular vs. islamist polarization in egypt on twitter," in *Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining*, 2013, pp. 290–297.
- [6] K. Darwish, W. Magdy, and T. Zanoluda, "Improved stance prediction in a user similarity feature space," in *Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining 2017*, 2017, pp. 145–148.
- [7] K. Darwish, P. Stefanov, M. Aupetit, and P. Nakov, "Unsupervised user stance detection on twitter," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 14, 2020, pp. 141–152.
- [8] B. Liang, Y. Fu, L. Gui, M. Yang, J. Du, Y. He, and R. Xu, "Target-adaptive graph for cross-target stance detection," in *Proceedings of the Web Conference 2021*, 2021, pp. 3453–3464.
- [9] S. Dutta, S. Caur, S. Chakrabarti, and T. Chakraborty, "Semi-supervised stance detection of tweets via distant network supervision," *arXiv preprint arXiv:2201.00614*, 2022.
- [10] D. Küçük and F. Can, "Stance detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 53, no. 1, pp. 1–37, 2020.
- [11] A. AlDayel and W. Magdy, "Stance detection on social media: State of the art and trends," *Information Processing & Management*, vol. 58, no. 4, p. 102597, 2021.
- [12] M. Hardalov, A. Arora, P. Nakov, and I. Augenstein, "A survey on stance detection for mis- and disinformation identification," *arXiv preprint arXiv:2103.00242*, 2021.
- [13] R. Villa-Cox, A. R. KhudaBukhsh, K. M. Carley *et al.*, "Exploring polarization of users behavior on twitter during the 2019 south american protests," *arXiv preprint arXiv:2104.05611*, 2021.
- [14] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," 2002.
- [15] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical review E*, vol. 76, no. 3, p. 036106, 2007.
- [16] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 1, pp. 55–67, 2007.
- [17] S. Gregory, "Finding overlapping communities in networks by label propagation," *New journal of Physics*, vol. 12, no. 10, p. 103018, 2010.
- [18] E. Colleoni, A. Rozza, and A. Arvidsson, "Echo chamber or public sphere? predicting political orientation and measuring political homophily in twitter using big data," *Journal of communication*, vol. 64, no. 2, pp. 317–332, 2014.
- [19] G. Kossinets and D. J. Watts, "Origins of homophily in an evolving social network," *American journal of sociology*, vol. 115, no. 2, pp. 405–450, 2009.
- [20] F. Alimadadi, E. Khadangi, and A. Bagheri, "Community detection in facebook activity networks and presenting a new multilayer label propagation algorithm for community detection," *International Journal of Modern Physics B*, vol. 33, no. 10, p. 1950089, 2019.
- [21] D. Liu, H.-Y. Bai, H.-J. Li, and W.-J. Wang, "Semi-supervised community detection using label propagation," *International Journal of Modern Physics B*, vol. 28, no. 29, p. 1450208, 2014.
- [22] X.-K. Zhang, C. Song, J. Jia, Z.-L. Lu, and Q. Zhang, "An improved label propagation algorithm based on the similarity matrix using random walk," *International Journal of Modern Physics B*, vol. 30, no. 16, p. 1650093, 2016.
- [23] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.
- [24] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The world wide web conference*, 2019, pp. 2022–2032.
- [25] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.
- [26] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.
- [27] J. Cañete, G. Chaperon, R. Fuentes, J.-H. Ho, H. Kang, and J. Pérez, "Spanish pre-trained bert model and evaluation data," in *PMLADC at ICLR 2020*, 2020.