APPENDIX

A. AUTOMATIC TEXT CLASSIFICATION METHODS

A.1 2-Pass Methods

Table 1 summarizes important information about the 2-pass approaches we compare here, highlighting the year, Basic Paradigm of the representation, input (words, documents and/or labels), if there is any type of statistical tests in the evaluation procedure of the original paper, if there is tuning and how it was done, and finally, if the input uses pre-trained word embeddings.

TFIDF. The simplest and almost universally used approach in which a document is represented as term vector with TFIDF weights of terms as elements. TFIDF_i of element i in the vector is given by $\text{TF}_i \times \text{IDF}_i$, where TF_i is the frequency of term i in the document and $\text{IDF}_i = log(\frac{N}{df_i})$, where N is total number of documents and df_i is the document frequency of the i-th term.

Metafeatures. Strategies based on Metafeatures (MFs) extract information from other more basic features (such as TFIDF) aiming at improving the feature space based on the main assumption that close documents tend to belong to the same class. The MFs we exploit here are the most effective ones according to [6]. We adopt the combined use of the similarity scores (cosine and $l2$) between a document and each category centroid as meta-features that exploit global information, and the similarity between a document and its neighbors from each category as meta-features that exploit local information. We also complement the aforementioned meta-features using the recently proposed ones that indicate hard-to-classify target documents [3]. Such meta-features evaluate the proportion of correctly classified (using the SVM classifier) neighbors of a target document, and the discrepancy between the classification of a target document and its neighbors. Our implementation of the MF approach was obtained from the authors of the original work, based on direct communications.

Predictive Text Embedding (PTE)[3]. PTE is a semi-supervised method to learn document representations. PTE fills a gap of previous unsupervised methods that do not consider the labeled information when learning the representation. The method learns distributed representation of text by embedding the heterogeneous text network into a low dimensional space. This network is composed of three bipartite sub-networks: word-word, word-document, and word-label, with word vertices, shared across the three networks.

Simple Word-Embedding based Models (SWEM). SWEM consists of parameter-free pooling operations, relative to word-embedding-based RNN/CNN models. In the original work, the authors conducted a point-by-point comparative study of pooling functions to construct the representation of a document based on the words vectors occurring in it. The studied pooling functions were: (1) Max-pooling; (2) Average-pooling; (3) An heuristic Hierarchical-pooling; and (4) concatenation of Max- and Average-pooling. In the original implementation[4], the authors construct document representations and then apply a Multilayer Perceptron to classify documents, thus being considered a 2-pass approach (SWEM Default). In addition to this approach, we conduct experiments with SWEM as an end-to-end (E2E) algorithm, where class prediction is made with the same fully connected layer used during training to build the representations.

Text GCN [7] is a text classification technique based on Graph Convolutional Networks (GCN) [5]. A graph connecting words and documents as nodes is constructed to represent the whole corpus. Edges represent the co-occurrences between words and the word/document associations. Edge weights between word nodes are obtained from the Pointwise Mutual Information (PMI) metric, while weights of word/document edges are based on TFIDF. The algorithm learns both word and document embeddings automatically, without the need of a pre-trained word embedding. After the corpus graph is created, a two-layer GCN is used to perform the textual classification. GCN [8] is a Multilayer Neural Network capable of operating directly on a graph and deducing node vectors based on node neighborhood information. Notice that this representation is extremely large, being very memory consuming. The authors of Text GCN tested it only in datasets with less than 20k documents.

A.2 End-To-End Methods

Table 2, on its turn, summarizes important information about the end-to-end approaches we compare here, highlighting the year, type of the architecture, input of the neural network (words, chars or sentences), if there is any type of statistical tests in the evaluation procedure, if there is tuning and how it was done, and finally, if the input uses pre-trained word embeddings.

FastText learns vectors for character n-grams found within each word, as well as for the complete word. At each training step in FastText, the mean of the target word and sub-word vectors are used for training. The adjustment that is calculated from the error is then used uniformly to update each of the vectors that were combined to form the target. This adds a lot of additional computation cost to the training step. The trade-off is a set of word-vectors that contain embedded sub-word information. In [9], the authors presented a supervised version of FastText word and document representation[6], which used the hierarchical softmax as loss function to approximates the softmax with a much faster computation. The idea is to build a binary tree whose leaves correspond to the labels. As an additional experiment, we also use document vectors as input to train a linear classifier, which was chosen SVM.

CNN. The convolutional architecture we exploit in our comparative work deals with document classification at character level using embeddings of characters [11]. This architecture showed excellent results for the ATC task in the authors' experimental evaluation [11, 16]. The CNN architecture is very straightforward – it contains nine deep layers (6 convolutional and 3 fully connected layers) considering 1024 kernel filters. Its character alphabet consists of 70 characters with feature length of 1014, with a dropout of 0.5 in between fully connected layers[14].

[3]Obtained from https://github.com/mnqu/PTE.

[4]Obtained from https://github.com/dinghanshen/SWEM.

[5]Our implementation was obtained from https://github.com/yao8839836/text_gcn.

[6]Our implementation of FastText was obtained from https://fasttext.cc/.

[14]. The Implementation of CNN, VDCNN and HAN are available at https://github.com/ArdalanM/nlp-benchmarks

Table 1: Research Papers: 2-Pass Approaches

Paper	Year	Paradigm	Input Level	Statistical Tests	Tuning
TFIDF [1]	1972	Statistical Information	Words, Documents	—	—
MetaFeatures [2, 3]	2018, 2019	Document Distances	TFIDF Document Vector	Yes	Tuning on validation set
PTE[1][4]	2015	Relational	Word, Document, Label	No	Tuning on validation set
SWEM[2][5]	2018	Simple Pooling of Embeddings	Word, Document, Label	No	Tuning on validation set

Table 2: Research Papers: End to End Approaches

Paper	Year	Architecture	Input Level	Statistical Tests	Tuning	Embeddings
LSTM [10]	2015	LSTM	Sentence	No	tuning of some hyperparameters	—
CNN[7][11]	2015	CNN	Char	No	Not mentioned	—
FastText[8][9]	2016	Skipgram	Word and Char	No	—	—
HAN[9][12]	2016	Attention	Sentence and word	No	Tuning on validation set; grid-search on the learning rate	Word2vec
VDCNN[10][13]	2017	CNN	Char	No	Not mentioned	—
BERT[11][14]	2019	Transformer	Sentence	No	Finetuning of some hyperparameters	—
XLNet[12][15]	2019	Transformer	Sentence	No	Fine-tuning of some hyperparameters	—
TextGCN [13][7]	2019	Graph Convolution Network	Word, Document, Label	No	—	—

VDCNN conneau2017very proposes is a very deep convolutional neural network for ATC, also at character-level, which is the first work to go deeper than six convolutional layers for sentence classification. VDCNN architecture is inspired in the idea of stacks of convolutional blocks, considering different sizes of filters and pooling and same sizes of kernel (kernel=3). In the experiments, different depths are explored. The authors observe that depths of 9, 17, 29 and 49 perform better than other depths. The training process of these very deep architectures is time consuming, even without any tuning of hyperparameters. Although, there are no statistical tests with replications, authors claim that the increase of the network depth improves performance in all datasets and that CNNs at character-level can be an effective method. We exploit the 29-deep architecture [15].

Hierarchical Attention Network (HAN) is applied to word and sentence levels, and can construct different document representation to recognize more and less important contents in text. They aim is to determine relevant sections by modeling interactions of the words rather than their presence in text. Thus, HAN includes two levels of attention mechanisms constructing the document representation: one at word level and one at the sentence level, allowing the model to pay more or less attention to individual words and sentences. For further information about how the architecture is built the reader is referred to the authors paper [12] for more details and understanding of the architecture.

Long Short Term Memory (LSTM) [17] is a special flavor of recurrent neural network (RNN) [18], widely used in tasks where there is temporal or contextual dependence. In ATC, each token x_i of a text $x = \{x_1, x2, \ldots, x_T\}$ is mapped to a vector representation (embedding) and put forward to an LSTM unit. Each LSTM unit i receives as input the token x_i, the previous unit hidden and the previous unit cell state. After processing all tokens, the last hidden state is used as input to a dense layer that predicts the text class [10].

BERT [16], one of our strongest baselines, is an end-to-end deep learning classifier composed of a bidirectional Transformer encoder with 24 Transformer blocks, 1024 hidden layers, and 340M parameters. The model is pre-trained with a 3.3 billion word corpus including BooksCorpus [17] (800 million words) and English Wikipedia (2.5 billion words). The original model run on 16 TPU pods for training. In short, it predicts missing words from a sentence. The authors proposed two pre-training tasks *Masked language model (MLM)* – this technique masks words with a probability of 15% and the model is trained to predict the masked words. and (ii) *Next sentence prediction (NSP)* – it trains the model to predict weather two sentences are consecutive or not. BERT uses a multi-layer bidirectional Transformer encoder whose self-attention layer acts forward and backwards. BERT redefined the state-of-the-art for 11 natural language processing task.

XLNet [19] is a recent E2E deep learning classifier that has outperform BERT in 20 natural language tasks [18]. BERT has the disadvantage of assuming that the predicted tokens are independent given unmasked tokens, a strong simplification in natural language. XLNet, a generalized autoregressive pre-training model, address this shortcoming by using a combination of advantages of autoregressive (AR) and autoencoder (AE) models. In other words, the method uses a permutation language modeling objective to combine the advantages of AR and AE methods. XLNet is pre-trained with the same parameters and word corpus as BERT. In addition, three word corpus are used, resulting in a 32.89B word corpus used to pre-training the model. Another notable difference is that training was performed with 512 TPU v3 chips.

B. DATASETS

We evaluate the effectiveness of the models on five large (more than 100,000 documents) scale document classification datasets [11] – AG's news corpus (AGNEWS), Sogou news corpus (SOGOU), Yelp Review Full, IMDB Reviews and Yahoo! Answers – and four smaller real-world textual datasets very well known in the ATC community – 20 News-

[15] 14

[16] Available in https://github.com/yaserkl/

[17] Available in https://googlebooks.byu.edu/

[18] Our implementation was obtained from https://github.com/zihangdai/xlnet

groups (20NG), WebKB (WebKB), Reuters (REUT), ACM Digital Library (ACM), respectively. The idea of separating in larger and smaller datasets is to analyze whether the analyzed methods behave differently on these two scenarios. This is important as ATC became pervasive in a multitude of applications and domains, many which are small or mid-sized (i.e., does not contain hundreds of thousands of documents). Table 3 describes the details for each dataset regarding the domain (documents' theme), size, number of features, class distribution, average document length and skewness.

- **AG's news corpus (AGNEWS)**: This AG's corpus of news articles was collecetd from the web[19]. The whole corpus contains 496,835 categorized news articles from more than 2000 news sources. Four largest classes (World, Sports, Business and Sci/Tech) from this corpus were chosen to construct the dataset used in the experiments, using only the title and description fields.

- **Sogou news corpus (SOGOU)**: It is a Chinese dataset from the combination of the SogouCA and SogouCS news corpora, containing 500K news articles in various topic channels. They were labeled by manually classifying their domain names. Five categories were defined: "sports", "finance", "entertainment", "automobile" and "technology". The models for English can be applied to this dataset without change. The fields used are title and content [20].

- **Yelp reviews full (YELP)**: It was obtained from the Yelp Data Challenge of 2015, containing about 700K reviews.

- **IMDB Reviews**: contains 348,415 user reviews about 50,000 movies. The scores for the movies, in a range [0,10], were discretized so that 10 classes are considered for classification. This is a highly imbalanced dataset.

- **Yahoo! Answers** (Yahoo! Answers Comprehensive Questions and Answers version 1.0). Zhang et. al. [20] collected this set of 4,483,032 questions and used their answers across the 10 largest main categories for build the classification dataset. The used fields include question title, question content and best answer.

- **4 Universities (4UNI), a.k.a, WebKB**: contains Web pages collected from Computer Science departments of four universities (Cornell (867 pages), Texas (827), Washington (1205), Wisconsin (1263) and 4,120 miscellaneous pages collected from other universities) by the Carnegie Mellon University (CMU) text learning group[20]. There is a total of 8,282 web pages, classified into 7 categories: "student", "faculty", "staff", "department", "course", "project" and "other". Table 3 shows 8199 instances. This is due to a preprocessing step which removed documents with a few words [21].

- **20 Newsgroups (20NG)** is a classical and popular dataset for experiments in text applications of machine learning techniques. It contains 18,846 newsgroups documents[21], partitioned almost evenly across 20 different newsgroups categories.

- **ACM-Digital Library (ACM)**: a subset of the ACM Digital Library with 24, 897 documents containing articles related to Computer Science. We considered only the first level of the taxonomy adopted by ACM, where each document is assigned to one of 11 classes.

- **Reuters (REUT)**: this is a classical text dataset, composed by news articles collected and annotated by Carnegie Group, Inc. and Reuters, Ltd. We consider here a set of 13, 327 articles, classified into 90 categories.

[19]http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

[20]http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data

[21]http://qwone.com/~jason/20Newsgroups/

C. REFERENCES

[1] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[2] S. Canuto, D. X. Sousa, M. A. Gonçalves, and T. C. Rosa, "A thorough evaluation of distance-based meta-features for automated text classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, pp. 2242–2256, Dec 2018.

[3] S. D. Canuto, T. Salles, T. C. Rosa, and M. A. Gonçalves, "Similarity-based synthetic document representations for meta-feature generation in text classification," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pp. 355–364, 2019.

[4] J. Tang, M. Qu, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1165–1174, ACM, 2015.

[5] D. Shen, G. Wang, W. Wang, M. R. Min, Q. Su, Y. Zhang, C. Li, R. Henao, and L. Carin, "Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Melbourne, Australia), pp. 440–450, Association for Computational Linguistics, July 2018.

[6] S. Canuto, D. X. Sousa, M. A. Gonçalves, and T. C. Rosa, "A thorough evaluation of distance-based meta-features for automated text classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, pp. 2242–2256, Dec 2018.

[7] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 7370–7377, 2019.

[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[9] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *CoRR*, vol. abs/1607.01759, 2016.

Dataset	Domain	Size	Class Distribution						Avg Doc. Length (words)	Skewness
			# Features	# Classes	Minor Class	Median	Mean	Major Class		
WEBKB	Web pages of CS Departments	8,199	23,047	7	137	926	1,171	3705	209	Imbalaced
REUT	News Articles	13,327	27,302	90	2	29	148	3964	171	Extremely Imbalanced
20NG	Newsgroups documents	18,846	97,401	20	628	984	942	999	296	Balanced
ACM	Paper titles	24,897	48,867	11	63	2,041	2,263	6,562	65	Imbalaced
AGNEWS	News Articles	127,600	39,837	4	31,900	31,900	31,900	31,900	37	Balanced
IMDB review	Movie Reviews	348,415	115,831	10	12,836	31,551	34,841	63,233	326	Imbalanced
SOGOU	News articles (Chinese)	510,000	98,974	5	102,000	102,000	102,000	102,000	588	Balanced
YELP Full	Place reviews	700,000	115,371	5	140,000	140,000	140,000	140,000	136	Balanced
Yahoo Answer	Questions and Answers	1,460,000	1,554,607	10	146,000	146,000	146,000.0	146,000.0	92	Balanced

Table 3: Datasets Statistics

[10] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 3079–3087, Curran Associates, Inc., 2015.

[11] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 649–657, Curran Associates, Inc., 2015.

[12] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 1480–1489, 2016.

[13] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, vol. 1, pp. 1107–1116, 2017.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "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)*, pp. 4171–4186, 2019.

[15] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *arXiv preprint arXiv:1906.08237*, 2019.

[16] Y. Xiao and K. Cho, "Efficient character-level document classification by combining convolution and recurrent layers," *arXiv preprint arXiv:1602.00367*, 2016.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[19] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," 2019.

[20] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *NIPS*, 2015.

[21] R. Campos, S. Canuto, T. Salles, C. C. de Sá, and M. A. Gonçalves, "Stacking bagged and boosted forests for effective automated classification," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, (New York, NY, USA), pp. 105–114, ACM, 2017.