POLITECNICO DI TORINO

MASTER THESIS

Semi-Supervised Machine Learning & Deep Learning Models in Crisis-Related Informativeness Classification

BY
ALESSANDRO RENNOLA
B.S. in Computer Engineering, Politecnico di Torino, Torino, Italy, 2017

Defense Committee:

Cornelia Caragea, Chair and Advisor
Elena Maria Baralis, Advisor
Erdem Koyuncu

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to the people without whom my thesis may not have been possible.

I would like to thank my main advisor Professor Cornelia Caragea, for the unceasing guidance and advice expressed throughout this ambitious project. This work is partially funded by Professor Caragea, to whom I reiterate my gratitude and appreciation.

I am indebted to my advisor Professor Elena Maria Baralis, who supported my choices and provided inestimable insights about this thesis. This thesis is a double degree project with Politecnico di Torino and she accepted to be my appointee to oversee this collaboration.

I would like to thank Professor Erdem Koyuncu for partaking in the Thesis Committee. His contributions are extremely valuable and appreciated.

I am thankful to my double degree program supervisors, Lynn Thomas and Jenna Stephens, for their supervision on the 'red tape' of bureaucracy.

Finally, I am especially grateful to my family. Their unwavering love and support have guided me through the hurdles and the complications of not only this thesis, but also my entire academic career. Special thanks my parents Michele and Maria, for their inspiration to pursue my dreams. You are the wind beneath my wings.

AR

TABLE OF CONTENTS

CHAPTER **PAGE**

TABLE OF CONTENTS (continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

NB	Naïve Bayes
SVM	Support-Vector Machine
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
Bi-LSTM	Bidirectional Long-Short Term memory
AL	Adversarial Loss
VAL	Virtual Adversarial Loss
RNN	Recurrent Neural Network
TF	Term Frequency
IDF	Inverse Document Frequency
EM	Emergency Manager
PIO	Public Information Officer
NIMS	National Incident Management System
PII	Personally Identifiable Information
ARC	American Red Cross
API	Application Program Interface
SA	Situational Awareness

LIST OF ABBREVIATIONS (continued)

EOS	End of Sequence
SGD	Stochastic Gradient Descent
SSL	Semi-Supervised Learning

CITED LITERATURE

Alam, F., Imran, M., and Ofli, F. (2019). Crisisdps: Crisis data processing services.

Alam, F., Joty, S., and Imran, M. (2018). Graph based semi-supervised learning with convolution neural networks to classify crisis related tweets. In *Twelfth International AAAI Conference on Web and Social Media*.

Ashktorab, Z., Brown, C., Nandi, M., and Culotta, A. (2014). Tweedr: Mining twitter to inform disaster response.

Baron, N. S. (2003). Language of the internet. *The Stanford handbook for language engineers*, pages 59–127.

Ben-Hur, A., Horn, D., Siegelmann, H. T., and Vapnik, V. (2001). Support vector clustering. *Journal of machine learning research*, 2(Dec):125–137.

Blum, A. and Mitchell, T. (1977). Combining labeled and unlabeled data with co-training.

Burel, G., Saif, H., Fernandez, M., and Alani, H. (2017). On semantics and deep learning for event detection in crisis situations.

Caragea, C., Bulgarov, F., and Mihalcea, R. (2015). Co-training for topic classification of scholarly data. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2357–2366.

Caragea, C., McNeese, N., Jaiswal, A., Traylor, G., Kim, H.-W., Mitra, P., Wu, D., Tapia, A. H., Giles, L., Jansen, B. J., et al. (2011). Classifying text messages for the haiti earthquake. In *Proceedings of the 8th international conference on information systems for crisis response and management (ISCRAM2011)*. Citeseer.

Chamasemani, F. F. and Singh, Y. P. (2011). Multi-class support vector machine (svm) classifiers – an application in hypothyroid detection and classification. In *2011 Sixth International Conference on Bio-Inspired Computing: Theories and Applications*, pages 351–356.

CITED LITERATURE (continued)

Chen, X. and Cardie, C. (2018). Unsupervised multilingual word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 261–270, Brussels, Belgium. Association for Computational Linguistics.

Conneau, A., Lample, G., Ranzato, M., Denoyer, L., and Jégou, H. (2017). Word translation without parallel data. *CoRR*, abs/1710.04087.

Cozman, F. G., Cohen, I., and Cirelo, M. C. (2003). Semi-supervised learning of mixture models. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 99–106.

Cresci, S., Tesconi, M., Cimino, A., and Dell'Orletta, F. (2015). A linguistically-driven approach to cross-event damage assessment of natural disasters from social media messages. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 1195–1200, New York, NY, USA. ACM.

Derczynski, L., Meesters, K., Bontcheva, K., and Maynard, D. (2018). Helping crisis responders find the informative needle in the tweet haystack. *arXiv preprint arXiv:1801.09633*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A. J., and Vapnik, V. (1997). Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161.

Ekstrm, R. J., Lunn, D., Jackson, C., Best, N., and Thomas, A. (2013). The bugs book: A practical introduction to bayesian analysis.

Ferrario, M. A. A., Simm, W., Whittle, J., Rayson, P., Terzi, M., and Binner, J. (2012). Understanding actionable knowledge in social media: Bbc question time and twitter, a case study. In *Sixth International AAAI Conference on Weblogs and Social Media.*

Florescu, I. (2014). *Probability and Stochastic Processes.* Wiley.

Fraustino, J., Liu, B., Jin, Y., for the Study of Terrorism, N. C., to Terrorism (U.S.), R., of Homeland Security. Science, U. S. D., and Directorate, T. (2012). *Social Media Use During Disasters: a Review of the Knowledge Base and Gaps.* National Consortium for the Study of Terrorism and Responses to Terrorism.

CITED LITERATURE (continued)

Gollapalli, S. D., Caragea, C., Mitra, P., and Giles, C. L. (2013). Researcher homepage classification using unlabeled data. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 471–482, New York, NY, USA. ACM.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. The MIT Press.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Grefenstette, E., Blunsom, P., de Freitas, N., and Hermann, K. M. (2014). A deep architecture for semantic parsing. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 22–27, Baltimore, MD. Association for Computational Linguistics.

Hearst, M. A. (1998). Support vector machines. *IEEE Intelligent Systems*, 13(4):18–28.

Hiltz, S. R., Kushma, J. A., and Plotnick, L. (2014). Use of social media by us public sector emergency managers: Barriers and wish lists. In *ISCRAM*.

Hiltz, S. R. and Plotnick, L. (2013). Dealing with information overload when using social media for emergency management: Emerging solutions. In *ISCRAM*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Hu, B., Lu, Z., Li, H., and Chen, Q. (2014). Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050.

Hughes, A., Palen, L., Sutton, J., Liu, S., and Vieweg, S. (2008). Site-seeing in disaster: An examination of on-line social convergence.

Hughes, A. L. and Palen, L. (2009). Twitter adoption and use in mass convergence and emergency events. *International journal of emergency management*, 6(3-4):248–260.

Hughes, A. L. and Palen, L. (2012). The evolving role of the public information officer: An examination of social media in emergency management. *Journal of Homeland Security and Emergency Management*, 9(1).

Imran, M., Castillo, C., Diaz, F., and Vieweg, S. (2015). Processing social media messages in mass emergency: A survey. *ACM Computing Surveys (CSUR)*, 47(4):67.

CITED LITERATURE (continued)

Imran, M., Castillo, C., Lucas, J., Meier, P., and Vieweg, S. (2014). Aidr: Artificial intelligence for disaster response. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 159–162. ACM.

Imran, M., Elbassuoni, S., Castillo, C., Diaz, F., and Meier, P. (2013a). Extracting information nuggets from disaster-related messages in social media. In *Iscram*.

Imran, M., Elbassuoni, S., Castillo, C., Diaz, F., and Meier, P. (2013b). Practical extraction of disaster-relevant information from social media. *WWW 2013 Companion - Proceedings of the 22nd International Conference on World Wide Web*.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML'98, pages 137–142, Berlin, Heidelberg. Springer-Verlag.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland. Association for Computational Linguistics.

Kendall, M. G. et al. (1948). The advanced theory of statistics. vols. 1. *The advanced theory of statistics. Vols. 1.*, 1(Ed. 4).

Kheradpisheh, S. R., Ghodrati, M., Ganjtabesh, M., and Masquelier, T. (2016). Deep networks can resemble human feed-forward vision in invariant object recognition. *Scientific Reports*, 6:32672.

Kiefer, J. and Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function. *Ann. Math. Statist.*, 23(3):462–466.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

CITED LITERATURE (continued)

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324.

Lambert, L., Moschovitis, C. J., Poole, H. W., and Woodford, C. (2005). *The internet: a historical encyclopedia*, volume 2. ABC-CLIO.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Lecun, Y. and Bengio, Y. (1995). *Convolutional networks for images, speech, and time-series.* MIT Press.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551.

Li, X., Liu, G., Ling, A., Zhan, J., An, N., Li, L., and Sha, Y. (2008). Building a practical ontology for emergency response systems. In *2008 international conference on computer science and software engineering*, volume 4, pages 222–225. IEEE.

Lindsay, B. R. (2011). Social media and disasters: Current uses, future options, and policy considerations.

Liu, S., Palen, L., Sutton, J., Hughes, A., and Vieweg, S. (2008). In search of the bigger picture: The emergent role of on-line photo sharing in times of disaster.

Ludwig, T., Reuter, C., Siebigteroth, T., and Pipek, V. (2015). Crowdmonitor: Mobile crowd sensing for assessing physical and digital activities of citizens during emergencies. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 4083–4092. ACM.

M Kendra, J. and Wachtendorf, T. (2003). Reconsidering convergence and converger legitimacy in response to the world trade center disaster. *Research in Social Problems and Public Policy*, 11:97–122.

MacEachren, A. M., Jaiswal, A., Robinson, A. C., Pezanowski, S., Savelyev, A., Mitra, P., Zhang, X., and Blanford, J. (2011). Senseplace2: Geotwitter analytics support for situational awareness. In *2011 IEEE conference on visual analytics science and technology (VAST)*, pages 181–190. IEEE.

CITED LITERATURE (continued)

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Marwick, A. E. and Boyd, D. (2011). I tweet honestly, i tweet passionately: Twitter users, context collapse, and the imagined audience. *New Media & Society*, 13(1):114–133.

McCallum, A. and Nigam, K. (1998). A comparison of event models for naive bayes text classification.

Mihalcea, R. (2004). Co-training and self-training for word sense disambiguation. In *CoNLL*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Chen, K., Corrado, G. S., and Dean, J. A. (2015). Computing numeric representations of words in a high-dimensional space. US Patent 9,037,464.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2017). Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Mitchell, T. (1999). The role of unlabeled data in supervised learning. In *Proceedings of the Sixth International Colloquium on Cognitive Science*.

Miyato, T., Dai, A. M., and Goodfellow, I. (2016). Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*.

Miyato, T., Maeda, S.-i., Ishii, S., and Koyama, M. (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*.

Nazer, T. H., Morstatter, F., Dani, H., and Liu, H. (2016). Finding requests in social media for disaster relief. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 1410–1413. IEEE Press.

CITED LITERATURE (continued)

Neppalli, V. K., Caragea, C., and Caragea, D. (2018). Deep neural networks versus naive bayes classifiers for identifying informative tweets during disasters. In *ISCRAM*.

Olteanu, A., Vieweg, S., and Castillo, C. (2015). What to expect when the unexpected happens: Social media communications across crises. In *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing*, pages 994–1009. ACM.

Palen, L., Anderson, K., Mark, G., Martin, J., Sicker, D., Palmer, M., and Grunwald, D. (2010). A vision for technology-mediated support for public participation & rescue in mass emergencies & disasters. *Proceedings of the 2010 ACM-BCS Visions of Computer Science Conference*.

Palen, L. and Anderson, K. M. (2016). Crisis informaticsnew data for extraordinary times. *Science*, 353(6296):224–225.

Palen, L. and Liu, S. B. (2007). Citizen communications in crisis: anticipating a future of ict-supported public participation. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 727–736. ACM.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pennington, J., Socher, R., and Manning, C. (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.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. cite arxiv:1802.05365Comment: NAACL 2018. Originally posted to openreview 27 Oct 2017. v2 updated for NAACL camera ready.

Rennie, J. D., Shih, L., Teevan, J., and Karger, D. R. (2003). Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 616–623.

CITED LITERATURE (continued)

Reuter, C., Hughes, A. L., and Kaufhold, M.-A. (2018). Social media in crisis management: An evaluation and analysis of crisis informatics research. *International Journal of Human–Computer Interaction*, 34(4):280–294.

Ruder, S., Vuli'c, I., and Sogaard, A. (2017). A survey of cross-lingual word embedding models.

Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition.

Schmidhuber, J. (2014). Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828.

Sharfuddin, A. A., Tihami, M. N., and Islam, M. S. (2018). A deep recurrent neural network with bilstm model for sentiment classification. In *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pages 1–4. IEEE.

Shen, Y., He, X., Gao, J., Deng, L., and Mesnil, G. (2014). Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, pages 373–374, New York, NY, USA. ACM.

Shklovski, I., Burke, M., Kiesler, S., and Kraut, R. (2010). Technology adoption and use in the aftermath of hurricane katrina in new orleans. *american Behavioral scientist*, 53(8):1228–1246.

Simm, W., Ferrario, M.-A., Piao, S., Whittle, J., and Rayson, P. (2010). Classification of short text comments by sentiment and actionability for voiceyourview. In *2010 IEEE Second International Conference on Social Computing*, pages 552–557. IEEE.

Subramanya, A. and Talukdar, P. P. (2014). Graph-based semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(4):1–125.

Varga, I., Sano, M., Torisawa, K., Hashimoto, C., Ohtake, K., Kawai, T., Oh, J.-H., and De Saeger, S. (2013). Aid is out there: Looking for help from tweets during a large scale disaster. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1619–1629.

CITED LITERATURE (continued)

Vieweg, S., Castillo, C., and Imran, M. (2014). Integrating social media communications into the rapid assessment of sudden onset disasters. In *International Conference on Social Informatics*, pages 444–461. Springer.

Yang, Z., Cohen, W. W., and Salakhutdinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861.*

Zade, H., Shah, K., Rangarajan, V., Kshirsagar, P., Imran, M., and Starbird, K. (2018). From situational awareness to actionability: Towards improving the utility of social media data for crisis response. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):195.

Zhang, S. and Vucetic, S. (2016). Semi-supervised discovery of informative tweets during the emerging disasters. *CoRR*, abs/1610.03750.

Zhang, Y. and Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820.*

Zhu, X. and Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–130.

SUMMARY

A study on the importance of the use of social media in times of emergency was carried out using a multilevel approach. Firstly, brief historical references were presented to contextualize and motivate the following work. Secondly, the most renown challenges to the adoption of social media crowdsourcing during crises were proposed. Furthermore, previous research was employed as a means to define informativeness and actionability.

The datasets and architectures used during this study are presented, together with the hyperparameters and the details of the experimental design.

The set of results is reported, along with a set of figures whose purpose is to graphically represent the performance of each model in every experimental instance. This work achieves a remarkable accuracy of 0.961 and 0.969 in, respectively, the English and the Italian datasets. A contextual analysis of the performance is carried out anticipating the final remarks and conclusions.

Finally, a section about future developments attempts to describe what could be the next steps in the topical research, possibly employing multi-language architectures, deep contextualized embeddings, hyperparameter tuning, data balancing techniques.

CHAPTER 1

ABSTRACT

This study examines the impact of several state-of-the-art Machine Learning and Deep Learning techniques in the context of semi-supervised disaster-related Twitter mining.

The goal is to create a model able to successfully classify informative tweets in the context of natural and human-induced disasters by employing several Machine Learning (Naïve Bayes and Support-Vector Machines) and Deep Learning (Convolutional Neural Networks, Bidirectional Long Short-Term Memory) mechanisms.

Firstly, we evaluate the performance of supervised instances. Subsequently, the supervised models are extended to assess the impact of semi-supervised techniques (self-training for NB, SVM, CNN; Virtual Adversarial Loss for Bi-LSTM). The accuracy of our Bi-LSTM model peaks at 0.961 in the English dataset, and 0.969 in the Italian dataset. In our knowledge, our semi-supervised learning models for informativeness classification outperform other supervised state-of-the-art models.

Finally, our conclusions are drawn as a means to provide a meaningful starting point for future research opportunities.

CHAPTER 2

INTRODUCTION

2.1 Purpose of the Study

This study examines the performance of several state-of-the-art Machine Learning and Deep Learning text classification models using supervised and semi-supervised techniques. The data upon which the classification techniques are built is part of a crisis-related tweet collection, labeled according to informativeness and relatedness to the disaster. This chapter aims at giving an overview of the analysis and the structure of the thesis.

2.2 Motivation

Since the dawn of life, people have always suffered from calamities in the form of personal and infrastructural damage and, although the improvements in engineering have allowed advances in damage reduction and expeditious resource dispatch, the damages could still be catastrophic.

People often seek ways to take part in the aftermath of a disaster, they may take photos, for informative, newsworthy or therapeutic purposes [Liu et al., 2008]. Alternatively, they may post messages on social feeds. Researchers categorize these different activities into displays of help, being anxious, returning, supporting, mourning, exploiting, and being curious, they often look for "evacuation routes, traffic conditions, plans to evacuate (...)". [Hughes et al., 2008] [M Kendra and Wachtendorf, 2003], [Palen et al., 2010].

Researchers have tried to list the different uses of social media during emergencies. According to [MacEachren et al., 2011], social media are used “To disseminate information to the public, to gather information from the public [Crowdsourcing], to monitor activities of crisis management professionals, as input to situational assessment for crisis management”.

Nowadays, Artificial Intelligence is one of the tools employed to support data analysis and the results in all the major areas of interest have such a great impact, that are redefining the baselines against which to compare the performance of, for example, empirical models. The data that is generated on social media has the virtue to be a valuable fit for Data Analysis, given its volume, the ease with which it can be collected (often) and the availability of targeted information across all the most common platforms. Specifically, Twitter, with more than 300 million monthly active users [1], has the peculiarity of being unfiltered and immediate in its tweets, therefore it is an excellent platform where to look for authenticity and up-to-date information [Marwick and Boyd, 2011].

Thus, it is clear that this knowledge could be helpful in Crisis Informatics: “a multidisciplinary field combining computing and social science knowledge of disasters; its central tenet is that people use personal information and communication technology to respond to disaster in creative ways to cope with uncertainty” [Palen and Anderson, 2016].

This project aims at comparing how different Machine Learning and Deep Learning architectures perform on informativeness classification, using Twitter as a content provider.

[1] https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/

2.3 Background

2.3.1 History of Social Media Usage During Emergencies

Internet is the global interconnection of networks whose expansion keeps its exponential pace for over three decades. Nowadays, Social Networks are incredibly popular, Facebook has reached 1.56 billion daily active users, on average, for March 2019 [2]. Twitter averages 126 Million daily active users in Q4 2018 [3].

The first situation in which people used the Internet is dated back in 1998, where protesters used to coordinate via newsgroups and emails [Lambert et al., 2005].

In 2001, we observe the first recorded case of disaster relief using social media crowdsourcing. [Reuter et al., 2018].

In 2003, a website was set up as a response to crises [Palen and Liu, 2007].

The next year, user-generated content was used in response to a crisis for the first time, when an electronic bulletin was set up and moderated for 10 days in response to the Indian Ocean Tsunami [Imran et al., 2015].

In 2005, during Hurricane Katrina, MySpace was used as a coordinator means for emergency responses with notable results [Shklovski et al., 2010].

[2] https://newsroom.fb.com/company-info/

[3] https://s22.q4cdn.com/826641620/files/doc_financials/2018/q4/Q4-2018-Shareholder-Letter.pdf

Twitter was used in 2007 during the wildfires that affected the vicinities of San Diego, California. Nate Ritter maintained a Twitter feed during the San Diego fires in 2007, you could monitor, provide and look for help by using the hashtag #sandiegofire [Hughes et al., 2008].

More recently, in 2012, the American Red Cross (ARC) summarized how social media and apps were effective during disasters [4]. Moreover, together with Dell, ARC launched the first-of-its-kind social media digital operations center for humanitarian relief with the goals of "source additional information from affected areas during emergencies to better serve those who need help; spot trends and better anticipate the public's need; and connect people with the resources they need, like food, water, shelter or even emotional support." [5].

[Imran et al., 2014] is a platform designed to perform automatic classification of crisis-related microblog communications. Starting from the principle of usefulness of small chunks of information, they build a model able to automatically tag unlabeled microblog posts to spot informative notions.

Nowadays, social media are being monitored to explore the physical and digital activities of crowds as a supplemental source of information. [Ludwig et al., 2015].

A more inclusive platform attempts a multi-level classification task using novel architectures: CrisisDPS [Alam et al., 2019]. The dimensions on which CrisisDPS operates are three-fold:

[4] https://www.prnewswire.com/news-releases/more-americans-using-mobile-apps-in-emergencies-168144726.html

[5] https://www.businesswire.com/news/home/20120307006328/en/American-Red-Cross-Dell-Launch-First-of-Its-Kind-Social

“disaster type classification: multi-label flagging to recognize earthquakes, fires, floods, hurricanes, bombings, shootings; informativeness: informative for humanitarian aid and response organizations to plan and launch relief efforts; humanitarian information type classification: affected individual, caution and advice, displaced and evacuations, donation and volunteering, infrastructure and utilities damage, injured or dead people, missing and found people, requests or needs, response efforts, and sympathy and support”.

2.3.2 Issues with Social Media Retrieval

When crawling social media messages, it is essential to keep in mind the end-users, those who “benefit from having curated information that describes a disaster or crisis and enhances situational awareness, including formal response agencies, members of the public” [Imran et al., 2015].

That is why it is necessary to provide access to well-structured information. As researchers, we must promptly process data. Systems are built to detect requests during disastrous events, making use of content and context of tweets. [Nazer et al., 2016].

The main challenges of event tracking and data extraction are, according to [Imran et al., 2015]:

- Inadequate Spatial and Temporal Information: Not every tweet carries geographical information.
- Noisy events: mundane events introduce noise, which is difficult for the algorithm to isolate and overcome.

- Describing the events: language seems to not constitute a grammatically well-formed description of the disaster.

Besides, social bots and fake news aggravate the possibility to extract useful information [Reuter et al., 2018]. Moreover, flagging sarcastic comments may be a challenge worthy of examination. It is particularly difficult to isolate these categories, they do not contribute to assessing the damages and their gravity.

Furthermore, social media offer content mining via APIs, but APIs “vary substantially from one platform to another, and also change over time” [Imran et al., 2015].

Finally, since social media activity grows exponentially during - mundane or disastrous - events, crawling and analyzing tweets during emergencies becomes unfeasible if not automated. This phenomenon is known as ’information overload’ [Hiltz and Plotnick, 2013]

2.3.3 Issues with Social Media Analysis

In [Hiltz et al., 2014], the authors interviewed US public sector emergency managers to define which are the current barriers to the use of social media and how those barriers can be overcome.

Emergency Managers (EMs) pose very compelling problems that retain them from using hundreds of potentially relevant posts. The goal of this work was to identify the issues that may occur during a large emergency.

Among the identified problems are issues of trustworthiness of information, lack of personnel time to work on information retrieval and gathering on social media, lack of regulatory instructions for its use and, also crucially important, regulations on how to ensure the privacy of users.

Information Systems need to allow structures and features for collecting, analyzing and sharing useful information during disasters, and government agencies need to come up with policies and standard procedures and protocols to integrate the systems in their crisis-response mechanisms. Furthermore, their employees must receive clear instructions on how to use the information systems to extract actionable knowledge [Hiltz et al., 2014].

[Lindsay, 2011] tries to assess what are the limitations and issues that characterize social media coverage during disasters. Social media have been used during emergencies, however, it is clear that “the number of personnel required to monitor multiple social media sources, and respond to and redirect incoming messages is also uncertain” and “responding to each message in a timely manner (...) may require an increase in the number of employees”.

Another critical examination is made by [Hughes and Palen, 2012], where they refer as misinformation as one of the most challenging tasks to overcome by Public Information Officers (PIOs), when dealing with citizens over social media. Public Information Officers (PIOs) are the public relations component of the National Incident Management Systems (NIMS).

Finally, the main challenges that data analysis must overcome as highlighted by [Imran et al., 2015] are:

- Scalability issues: each tweet is around 4KB when metadata is included.
- Content issues: “language in short blogs is fragmentary, laden with typographical errors, often bereft of punctuation, sometimes incoherent” [Baron, 2003].

- Privacy issues: social media may carry Personally Identifiable Information (PII) such as the location of users, therefore, must be carefully analyzed.

[Li et al., 2008] developed “a practical ontology for the response phase of standard mitigation, preparedness, response, and recovery process”. The research focuses on the semantic concepts of response preparation, emergency response, emergency rescue, aftermath handling.

2.3.4 Why Twitter?

With more than 320 Million active users, Twitter is 12th in the statistic of most widespread social media platforms and “the most widely used microblogging application” [MacEachren et al., 2011] [6] [7]. Its ease of use allows people to adopt it as a way of microblogging, documenting day-to-day life as well as sporadic events of interest.

Thus, during crises, people tend to use Twitter as an information source and use microblogging to broadcast disaster-related notions [Hughes and Palen, 2009].

Many responders have hence begun to incorporate Twitter messages and other social media as a way to monitor communications and calls for help. A part of these messages is extremely valuable, it is estimated that about 20% of the tweets describe content provided by affected individuals, which is not present in mainstream media [Olteanu et al., 2015].

Twitter is also popular among scientists because it is easier to obtain public data using Twitter APIs than from other sources. Moreover, tweets are easier to store, process and analyze

[6] https://www.fastcompany.com/90256723/twitters-q3-earnings-by-the-numbers

[7] https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

than other social media because of the limitation on the number (140) of characters [Reuter et al., 2018].

However, there are some drawbacks caused by the particularly short length of the messages. In fact, researchers argue that information extraction on microblogs is more difficult than performing the same analysis on articles [Imran et al., 2013b].

2.3.5 Why Semi-Supervised Learning?

Now that the integration of social media messages has been justified, we would like to motivate the choice of using semi-supervised learning algorithms.

It is crucially important to obtain assessments of the damages during the first hours of a crisis because Humanitarian Organizations, Public Information Officers and Emergency Responders can better coordinate the reaction to limit the infrastructural damages, the number of injured people, and, possibly, the number of casualties [Varga et al., 2013], [Vieweg et al., 2014].

Thus, relying on informative tweets in the early hours of a disaster would be an extremely valuable task, albeit particularly hard to obtain. The scarcity of labeled data during the initial phases of a disaster could prevent responders from obtaining valuable knowledge and allocating resources [Alam et al., 2018] . On the contrary, unlabeled data is fairly simple to obtain, thousands of tweets are posted every second [8].

Semi-supervised learning algorithms can be adopted to mitigate this problem. There are several models that could fall into this category. Among the most popular we can find co-

[8] https://www.internetlivestats.com/twitter-statistics/

training [Mitchell, 1999], self-training [Mihalcea, 2004], and, more recently, graph-based models [Subramanya and Talukdar, 2014].

It is only in recent times that semi-supervised techniques have been employed in the context of disaster-related microblogs classification. [Zhang and Vucetic, 2016] proposes to analyze unlabeled messages to extract word clusters, this information will serve as additional features for text classification. [Alam et al., 2018] adopts an inductive graph-based [Subramanya and Talukdar, 2014], [Yang et al., 2016] deep learning approach to classify relatedness to a disaster. Both approaches achieve significant improvements as compare to the supervised learning experiments.

CHAPTER 3

A REVIEW OF INFORMATIVENESS AND ACTIONABILITY

3.1 Informativeness

A crucial front of the state-of-the-art research is to characterize what informativeness and actionability are, why they are interesting topics to explore and what kind of information they can convey during emergencies.

3.1.1 Definition

A very interesting point from the paper [Derczynski et al., 2018] is that the task of flagging informative text is arduous for annotators if the definition does not comply with policies of simplicity and clarity. Furthermore, as it turns out, relevance and informativeness are deeply context-dependent and require very accurate analysis to be determined. Relevance must be indicative of a connection to the disaster in the matter. Informativeness must somehow reflect the presence of some kind of information, typically, damages to infrastructure or people.

For these reasons, after examining some of the alternatives, [Derczynski et al., 2018] came up with the following definitions: “Informative: the information in the message would be helpful with saving lives or assisting the authorities/other response teams in dealing better with the incident. Somewhat Informative: the message contains useful information that helps to understand better the situation. But no emergency or urgent information can be found in the tweet. Not Informative: the message does not contain useful information regarding the incident, but is instead expressing an opinion or contains unrelated information".

Furthermore, [Alam et al., 2019] flag a message as informative if "contains some useful information for humanitarian aid: warnings, cautions, reports about injured dead affected people,asking of offering rescue, volunteering, donations, reporting damaged houses, damaged roads (...)".

Finally, [Imran et al., 2013a] defines informativeness as "the message is of interest to other people beyond the authors immediate circle".

3.1.2 Contribution

Researchers have tried to extract informative content analyzing Twitter. [Caragea et al., 2011] approaches an automated classification of messages via manually generated labels and learning algorithms, key to delivering effectively the information to the Emergency Response Sector, which has the means to address the most urgent needs of the affected.

[Neppalli et al., 2018] compares Naïve Bayes, Convolutional Neural Networks, Recurrent Neural Networks and suggest that CNNs outperform all other methods.

Finally, [Alam et al., 2019] classifies informativeness on different datasets.

[Neppalli et al., 2018] and [Alam et al., 2019] work on CrisisLexT26; considering the magnitude and the importance of this dataset, we decided to perform our analysis on this collection.

3.2 Actionability

Actionable data is inestimable during the first hours of a disaster, because it can be used in practical efforts for save human lives by Emergency Responders and Humanitarian Organizations. They have the possibility to take action against perilous situations, but reacting effectively to emergencies is complicated, especially because the information available in the early hours of a disaster is anecdotal and fragmented.

Nevertheless, Many social media messages communicate during emergencies convey timely, actionable information [Imran et al., 2015] and Twitter is becoming more and more used as a media where to scout for actionable information and make impactful decisions [Olteanu et al., 2015].

3.2.1 Definition

According to [Ferrario et al., 2012] "little clarity exists in relation to what actionable knowledge is, whether it can be measured and where it is more likely to be found".

[Simm et al., 2010] defines actionability as "how actionable statements could provide a clear suggestion on how a product or a service could be improved". In Crisis Informatics, this definition could be extended to account for the need to provide a timely reaction to a particular situation.

3.3 Actionability versus Informativeness

Situational awareness and informativeness can be very interesting, yet general, concepts; hence, to integrate social media messages in response mechanisms, we need to signal information that can be acted upon. Previously, most of the definitions of actionability revolved around information that necessitates a response. However, in order for responders to incorporate tweets in disaster management, it may be necessary to extend the definition depending on the role of the responder. Researchers point out that references about the location and the time of the situation, the availability of resources, the trustworthiness of the source, the context of the disaster could prevent or allow responders to mobilize resources [Zade et al., 2018].

Considering the definitions and the current literature, our work considers the concept of informativeness and compares the obtained results with [Olteanu et al., 2015], [Alam et al., 2019], [Imran et al., 2014], [Neppalli et al., 2018], [Ashktorab et al., 2014]. We speculate that, in the future, a contribution of this nature, but traversing the concept of actionability, could be extremely valuable.

CHAPTER 4

MODELS

4.1 Supervised Machine Learning

4.1.1 Naïve Bayes Classifier

Naïve Bayes Algorithms are a collection of machine learning models used for classification purposes. These models employ existing features (or attributes) to assign a value to the class variable, which is the goal of the discriminative task. These techniques are named after Thomas Bayes, who first came up with a mathematical formulation for probability inference.

Bayes Theorem

Bayes' theorem can be mathematically represented as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{4.1}$$

[Kendall et al., 1948]

The probability of event A to be true given that B occurred (conditional probability of A given B) coincides with the "conditional probability of B given A multiplied by the marginal probability of A and divided by the marginal probability of B" [Ekstrm et al., 2013]. The marginal probability of X is defined as the probability of X independently of other events.

In other terms, the posterior probability (A given B) is equivalent to the prior (B given A) multiplied by the likelihood of A and divided by the evidence of B occurring.

$$posterior = \frac{prior * likelihood}{evidence} \tag{4.2}$$

Naïve Hypothesis

Naïve Bayes classifiers assume that all the features share the property of mutual conditional independence. Two events are independent if and only if their joint probability is equal to the product of their probabilities: [Florescu, 2014]

$$P(A, B) = P(A)P(B) \tag{4.3}$$

Such an assumption is often considered too simplistic, especially for cases in which the features are not actually independent. Nevertheless, Naïve Bayes systems can work surprisingly well, even when the conditional independence assumption is clearly violated [Russell and Norvig, 2009].

Classifier

The goal of the classifier [McCallum and Nigam, 1998] is to assign the most probable class label given the feature model:

$$y = \underset{j}{\operatorname{argmax}}\, P(C_j) \prod_{i=1}^{n} P(x_i|C_j) \tag{4.4}$$

In this case C_j is the j^{th} class, x_i is the i^{th} training data point.

Naïve Bayes classifiers for Text Classification require a set of examples for each class in which we wish to categorize text. The most commonly used algorithms for this purpose are Multinomial Naïve Bayes and Bernoulli Naïve Bayes.

Multinomial Naïve Bayes

$$y = \underset{j}{\operatorname{argmax}} P(C_j) \prod_{i=1}^{n} P(t_i|C_j) \tag{4.5}$$

$P(t_i|C_j)$ is the conditional probability of a term occurring in a document of class C. It is computed by counting how many times t appears in documents of class C, and dividing by the total occurrences of each token in documents of class C.

$$P(t|C) = \frac{T_{Ct}}{\sum_{t'} T_{Ct'}} \tag{4.6}$$

If one term had frequency equal to 0, then the probability of the whole product would reach 0. To eliminate zeros it is common practice to use add-one (Laplace) smoothing.

$$P(t|C) = \frac{T_{Ct} + 1}{\sum_{t'} T_{Ct'} + |V|} \tag{4.7}$$

By adding 1 to the numerator we avoid $P(t|C)$ to be 0, then we re-scale the results adding $|V|$ (size of the vocabulary) to the denomination. The latter step is required to re-normalize the summation of the term factors, it is immediate to demonstrate that $\sum_i P(t_i|C) = 1$.

Multivariate Bernoulli Naïve Bayes

In the Multivariate NB, or Bernoulli NB approach each word in the dictionary constitutes a feature in the feature model. Every feature is either True (1) if the word appears in that specific document of False (0) if it does not.

This model defines $P(t|C)$ as the fraction of the documents containing term t belonging to class C. It is still encouraged to smooth the probabilities: adding 1 to the numerator and 2 to the denominator avoids numerators dropping to 0 and causing the whole product to reach 0.

$$P(t|C) = \frac{T_{Ct} + 1}{\sum_{t'} T_{Ct'} + 2} \tag{4.8}$$

It is crucial to notice that, in this model, the absence of a term is considered indicative as its occurrence because, if a term occurs, it is going to be a part of the product as $P(t|C)$, if not part of the document, its corresponding term would be $(1 - P(t|C))$. Another critical difference between the two methods consists in how multiple occurrences are modeled, while with Multinomial NB $P(t|C)$ keeps track of the number of occurrences, Bernoulli NB, $P(t|C)$ models solely the occurrence/nonoccurrence, without accounting for the term frequency within a document.

Accordingly, this usually causes Multinomial NB to outperform Bernoulli NB when working with very long text instances, in fact, multiple occurrences could be a way to direct the classification task towards a more representative class instance.

Model Design

The following section focuses on Naïve Bayes for Text Classification problems. The first step of the Pipeline preprocesses the dataset using an NLTKPreprocessor and consists of:

- lower(). Converts the string to lower case.
- strip(). Removes all the whitespace characters from the beginning and the end of the word. We also strip '_' and '*'.
- stopwords(). Removes all the stopwords for the specified language. In our case, English or Italian are alternatively passed to the NLTKPreprocessor.
- stem(). Reduce the word to its word stem by removing prefixes and suffixes.

Secondly, we need to assign numerical values to the words, this way we can build a numerical model associated with each document. The most straightforward approach consists of computing the occurrences of the words in the document. Even though we do not have particularly large documents, this could still assign larger weights to larger documents. TF (Term Frequency) obviates this issue, normalizing the values using the number of total words in the document as a normalizing factor. Finally, we further modify the weighting system to account for IDF (Inverse Document Frequency) scores. By multiplying TF by the IDF score we penalize those terms that are most common in the text.

$$TF_{i,j} = \frac{n_{i,j}}{|d_j|} \tag{4.9}$$

Term Frequency corresponds to the number of words i in document j divided by the number of total words in document j.

$$IDF_i = \log(\frac{|D|}{|d : i \in D|}) \tag{4.10}$$

Inverse Document Frequency is given by the log of the division between the total number of documents and the number of documents in which i occurs.

$$TF - IDF_{i,j} = \frac{n_{i,j}}{|d_j|} \log(\frac{|D|}{|d : i \in D|}) \tag{4.11}$$

Now that we have a weighting system that works better than simple occurrence counting, we can apply it to our Multinomial Naïve Bayes algorithm [Rennie et al., 2003]. This is the last element of our Pipeline.

We employ Laplace smoothing of the probabilities (alpha=1.0) and let the prior class probabilities adjust according to the data (class_prior=None).

4.1.2 Support-Vector Machines

Support-Vector Machine is a vector space based Machine Learning model [Manning et al., 2008] mainly used for classification. There are extensions to the algorithm suitable for regression

(Support-Vector Regression Machines [Drucker et al., 1997]) and clustering (Support-Vector Clustering [Ben-Hur et al., 2001]) purposes. The goal of the classification task is to find the hyperplane that:

- Best separates the classes.
- Maximizes the distance between the training points and the decision boundary itself.

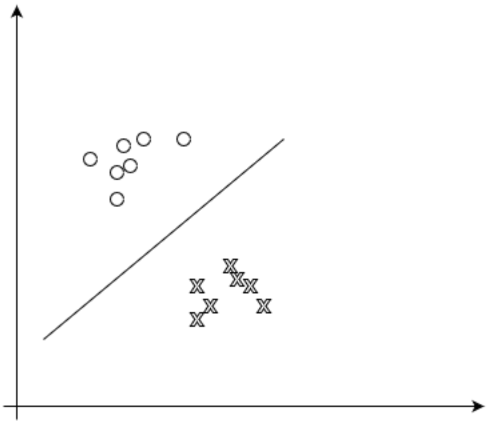


Figure 1. Linear Binary SVM classifier.

The goal of the SVM is to maximize the distance from the closest data points to the hyperplane. This distance is called margin of the SVM and the points whose distance to the separator corresponds to the margin are called support vectors. They can be referred as vectors because the data points form a vector with the origin. The hyperplane in Figure 1 divides the space in two sections, members of each side belong to the same class.

Properties

Support-Vector Machines often perform fairly well, they may outperform other techniques in situations with inadequate training data [Manning et al., 2008] and are especially useful for linearly separable data with a convex cost function, in which case the SVM always converges to a global optimum.

In its original version, SVM was introduced to solve a binary classification task with linearly separable classes. However, there are some valuable extensions to the algorithm that tackle multi-class discrimination and non linearly separable data.

Formal Description

Suppose d_i the desired class, $d_i = 1$ if the point belongs to class C_1 and $d_i = -1$ if the point belongs to class C_2. Let us define a weight vector w and a bias θ such that w is orthogonal to the hyperplane and:

$$\begin{aligned} w^T x_i + \theta \geq 0 \qquad & d_i = 1 \\ w^T x_i + \theta < 0 \qquad & d_i = -1 \end{aligned} \tag{4.12}$$

The linear classifier is then defined as:

$$f(x) = sign(w^T x_i + \theta) \tag{4.13}$$

the distance between the point and the separator becomes:

$$r = d_i \frac{w^T x_i + \theta}{|w|} \tag{4.14}$$

From this constraints we extract a quadratic function. Thus, the problem becomes a quadratic optimization problem with linear constraints. The quadratic equations are written as:

$$\max_{\alpha_i} \sum_{i}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j x_i^T x_j \tag{4.15}$$

And the constraints:

$$\textstyle\sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \, for \, all \, 1 \leq i \leq N$$

The solution assigns a 0 to all the α_i that are not support vectors. All the non-zero values correspond to support vectors.

Soft Margin Classification

If the data in the training set is not linearly separable, it is impossible for the SVM to produce a hard decision margin. In order to account for this phenomenon, the classifier will produce a 'soft' decision margin, allowing some points to be 'on the wrong side' of the boundary. These entries, considered as outliers, are characterized with a misclassification cost, proportional to the distance between the data point and its corresponding class [Chamasemani and Singh, 2011].

This procedure is realized using “slack variables ξ_i. A non-zero value for ξ_i allows x_i to not meet the margin requirement at a cost proportional to the value of ξ_i” [Manning et al., 2008].

Nonlinear SVM

One of the most common ways to solve the non-linearity of the classes is to map the data onto a higher dimensional space, called feature space. Typically, $p >> d$: the dimension of the feature space is much higher than the dimension of the original space. Now, the only thing left to do is to build a linear SVM classifier on the feature space.

It is common practice to define $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ and use this declaration as:

$$\max_{\alpha_i} \sum_{i}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j k(x_i, x_j) \tag{4.16}$$

To map x_i from the original space to the feature space we need to apply the transformation $\Phi : x \rightarrow \phi(x)$. For each i, j the dot product between x_i and x_j is mapped to $\phi(x_i)^T \phi(x_j)$ onto the feature space and such formulation is equivalent to $k(x_i, x_j)$ as previously defined.

This definition of k prevents the classifier from learning from a non-linear model. It is still necessary to map the points to a higher dimensional space, but we do not find the non-linearity at this time, resulting in a dot product between higher dimensional vectors. This is what is usually called ‘kernel trick’ and k is called ‘kernel function’.

Mercer’s Theorem

If a candidate function k is continuous, symmetric, and have a positive definite gram matrix, it is a valid kernel function.

SVM for Text Classification

SVMs are also used for text classification purposes SVM [Hearst, 1998]. Each word in the vocabulary is considered as a different feature in the feature space. Each document is represented in a N-dimensional space, where N is the number of features that constitute the model. The next step consists in obtaining a vectorized representation for the document, the vector is typically initialized as a term frequency counter for all the words in the vocabulary. Other vectors could be computed using $tf - idf$ or smoothing techniques.

Model Design

The initial processing is described in Section 4.1.1. NLTKPreprocess is in charge of the data cleansing, stopword removal and stemming.

The second step is modifying the weighting mechanism to prevent a bias towards large documents and very common words. Penalizing on common words is not strictly necessary since we remove language-specific stopwords, but TF-IDF is still interestingly resourceful as a weighting factor. An in-depth description of the weighting options can be found in section 4.1.1.

Now that the data is preprocessed and weighted, we can build an SVM classifier [Pedregosa et al., 2011].

The architectural choice was initially a 'canonical' SVM but the alternative that we propose is an SVM trained with Stochastic Gradient Descent (SGD) [Kiefer and Wolfowitz, 1952]. It is a very efficient approach to discriminative learning of linear classifiers such as SVM and Logistic Regression [1].

The 'hinge' loss function builds a soft-margin linear SVM, on the other side, 'modified_huber' uses a smoothed version of the hinge loss to increase tolerance to outliers. The latter is used for our SGD Classifier, it would be interesting to build different kernel SVM and cross evaluate their performance, this may be explored as future work.

L2 regularization is introduced to avoid overfitting, L1 could be also an interesting choice thanks to the property of feature selection which vacates the features deemed useless [Schmidhuber, 2014].

The constant that multiplies the regularization factor is named 'alpha'. The value is 0.001, other values such as the default 0.0001 were initially used with similar performance. We leave for future work this kind of hyperparameter optimization.

Setting the 'learning_rate = optimal' causes the value of alpha to coincide with the value of the learning rate.

[1] https://scikit-learn.org/stable/modules/sgd.html

4.2 Supervised Deep Learning

4.2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of multilayer perceptrons especially useful for pattern recognition applications such as image processing, video analysis and natural language processing [LeCun et al., 1989]. They are characterized by translation invariance and robustness to various forms of distortion like scaling and skewing. A crucial point of this architecture is that neighboring neurons share portions of the receptive field, combining percepts allows the extraction of general patterns. CNNs are identified by constraints which characterize the structure of the network [Lecun and Bengio, 1995].

- Feature extraction: each layer in the Neural Network contains perceptrons with the purpose to identify (hidden) patterns or features and extract them.
- Feature mapping: Each layer of the network contains multiple feature maps. The neurons in the layer are constrained to extract the features despite sharing the same set of weights.
  - shift (translation) invariance if using a small kernel size and a sigmoid function.
  - reduction of the number of free parameters by weight sharing.
- Subsampling: In order to reduce the impact of distortion, each convolutional layer is followed by a pooling layer which decreases the spatial resolution, in other words, decreases the size of the feature map.

Convolutional Layer

The convolution emulates the behavior of the human eye [Kheradpisheh et al., 2016]. Let us define l as the current layer of operation (l_0 the input data point), given d dimension of the tensor in the l^{th} layer, a filter is defined as a weight matrix of size $f * f$, with $2 \leq f \leq d - 1$.

Starting from the top-left corner of the l^{th} layer we perform a matrix multiplication operation between the filter and the portion of the image over which the filter is hovering. Subsequently, the filter is shifted to the right by a value named stride value s, until the right corner is reached. When this occurs, the filter is brought back to the left and shifted down by s positions. The process is repeated until the kernel (filter) is in the bottom-right corner of the image.

In most cases we use different feature mappings to analyze the image, each feature mapping is also called channel and they are used to recognize different patterns from the data.

It is possible to modify the input by adding a border of zeros around the image. This mechanism is often employed to improve the information gain at the borders. Furthermore, it is also possible to specify the size of the padding, usually, a very large padding size dilutes too much the information and is not necessarily desirable.

In picture Figure 2 we can see the result of the convolution between the input and the specified filter. The input is a 4×4 matrix, containing the weights for the specified channel. The filter is a 2×2 matrix. In this example, we use a value of 1 for the stride and no zero-padding is applied. If the number of channels is larger than one, the filter is applied to each channel and the resulting matrices are added together.

Pooling Layer

Pooling Layers aim at decreasing the size of the feature map, reducing the computational and statistical burden on the next layer. They are often added after a convolutional layer and help

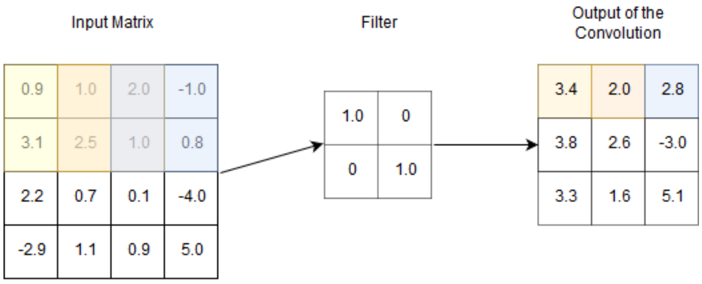


Figure 2. Output of a sample Convolutional layer.

"make the representation approximately invariant to small translations of the input. Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change" [Goodfellow et al., 2016].

The main pooling functions used in the pooling layers are:

- Max Pooling: Find the maximum for each portion of the feature map.
- Average Pooling: Average (L1-norm, L2-norm) over the portion of the rectangular neighborhood defined by the filter.
- Weighted Average Pooling: based on the center of the filter.

CNNs for Text Classification

The purpose of this section is to define how Convolutional Neural Networks can be used for text classification. It is common use to employ Convolutional Neural Networks classify visual data: images and, with some extensions, videos.

Nevertheless, they also find a valuable contribution when applied to Natural Language Processing. Classic NLP applications include semantic parsing [Grefenstette et al., 2014], search query retrieval [Shen et al., 2014], named entity recognition [Lample et al., 2016], sentence modeling [Kalchbrenner et al., 2014], semantic matching [Hu et al., 2014], text classification [Kim, 2014].

The first layer of the CNN embeds words in each document into weight vectors. The next set of layers is a series of convolutional and pooling layers with multiple filters. Then, it is useful to add a set of fully connected layers with (or without) dropout regularization. The final layer of the Neural Network is a softmax layer, customary to emulate a probability distribution.

It is possible to train the neural network starting from various embedding representations. Some of the most used embeddings are Word2Vec [Mikolov et al., 2013a], fastText [Joulin et al., 2017], GloVe [Pennington et al., 2014], ELMo [Peters et al., 2018], Bert [Devlin et al., 2018]. Some of these methods define a standard architecture to use to generate word vectors starting from the training data, other embeddings provide a set of pre-trained weight vectors of fixed size.

Model Design

Our implementation of the convolutional neural network follows the baseline of [Zhang and Wallace, 2015].

The sentence is tokenized into its words, each word is preprocessed and vectorized into its embedding. The tokenization includes the removal of special characters, basic suffixes and contracted verbal forms, removal of exclamation marks, question marks, and other punctuation marks. Finally, we remove parentheses and we convert the string to lowercase.

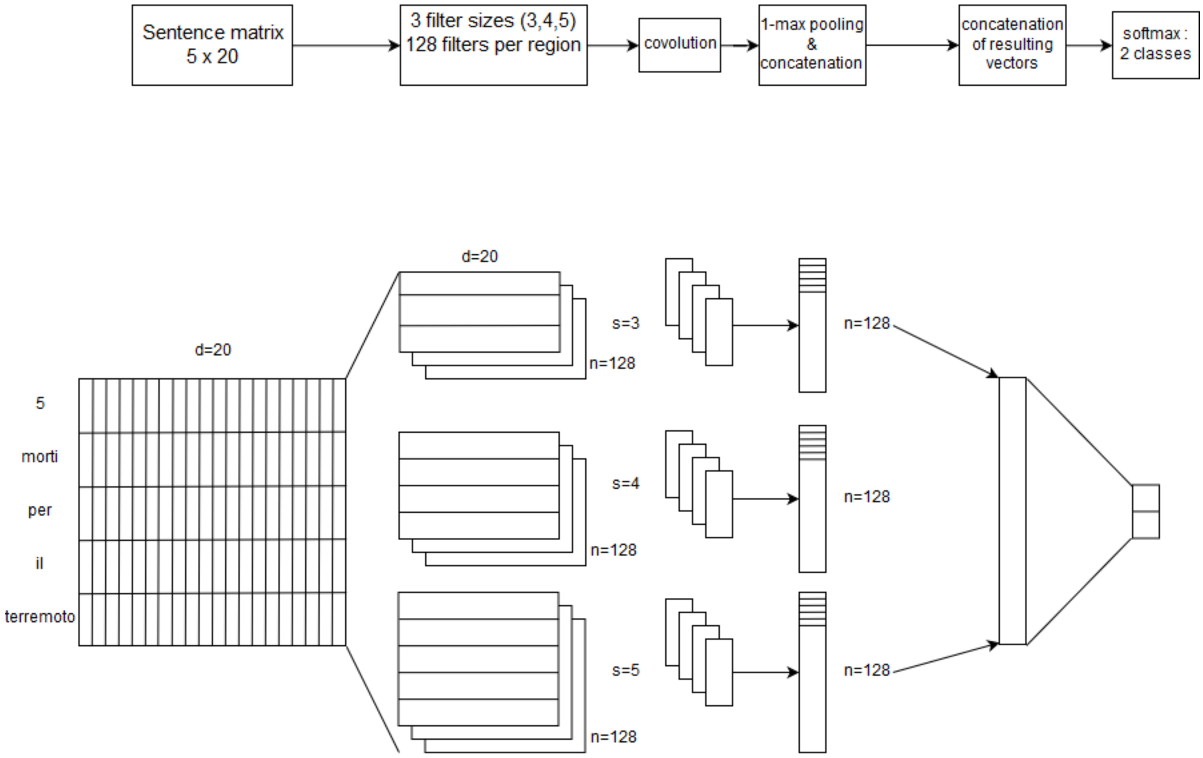


Figure 3. Diagram of a CNN for sentence classification.

The following explanation refers to CNN with multiple filters and varying filter-sizes for text classification in Figure 3.

We build 128 filters for each of the 3 filter sizes (3,4,5). We build a convolutional layer using the defined filters. The dimension of the filter is, therefore (3,4,5) x d (embedding dimension), the convolution moves in a temporal matter accordingly, starting from the first (3,4,5) words and moving towards the end of the document with a stride of 1. Filters generate feature maps of variable length [Zhang and Wallace, 2015].

Now we build a max-pooling layer over each of the convolved filters. Thus resulting in series of univariate vectors.

These features are concatenated to form a unique vector. Two neural units define the classes, applying the softmax function outputs two probabilities, one for each output class. Finally, the $\arg\max$ function outputs the most confident label. This is a binary classification task that outputs two values, they refer to the 'informative' and 'non-informative' classes.

Word2Vec

Word2Vec is the first of the word embeddings described in this section. It was created by Tomas Mikolov and his team at Google and patented in 2015 [Mikolov et al., 2015].

The goal of the technique is to represent words in high-dimensional vectors with the expectation that similar words tend to be close to each other [Mikolov et al., 2013a]. Furthermore, words can have multiple degrees of similarity [Mikolov et al., 2013b].

The most famous models are Continuous Bag of Words (CBOW) and Continuous Skip-Gram.

CBOW uses the context of a token to predict it, by building windows of specified length. The order in which the words are specified is not taken into account, their occurrence or non-occurrence is the discriminative factor.

Skip-grams, on the other side, use the current word to try to characterize the contextual window of tokens, closer words have a larger impact when building the weight vectors. In other words, the goal is to maximize the likelihood of observing a given context given the current word.

The first step of the algorithm is to create a model file using the list of words in every text snippet. The model is hence created with a vector dimension of 300 and trained on the documents in the corpus.

Now, we define the model to employ, for our purposes the training algorithm is Continuous Bag of Words (CBOW) 'sg=0' with averaged contexts of word vectors 'cbow_mean=1'.

Next, we define the window size 'window=5'. We keep all the words, even the ones with frequency of 1 'min_count=1'. Then, we set the worker threads to 4 'workers=4'.

Finally we set the learning rate to 0.025 'alpha=0.025' and its linear decay to 0.0001 'min_alpha=0.0001'.

fastText

fastText is a library for efficient learning of word representations and sentence classification. Its code and resources can be found in their GitHub page [2].

It is a “new approach based on the skip-gram model, where each word is represented as a bag of character n-grams. A vector representation is associated with each character n-gram; words being represented as the sum of these representations” [Joulin et al., 2017]. It serves as a new view on the data, with each token split into its subwords. The resulting n-grams are

[2] `https://github.com/facebookresearch/fastText`

embedded, the resulting word vectors are summed to constitute the embedding of the initial word.

We use pre-trained word vectors trained on 600B tokens on the Common Crawl corpus. The embedding can be found at [3]. Each word is represented in a 300-dimensional vector, it is trained with subword information on unlabeled documents.

The first line contains the number of words in the vocabulary and the size of vectors [Mikolov et al., 2017].

```
$ wget https://dl.fbaipublicfiles.com/fasttext/vectors-english/crawl-300d-2M
    -subword.zip
```

GloVe

"GloVe is an unsupervised learning algorithm for obtaining vector representations for words. 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" [Pennington et al., 2014] [4].

Our pre-trained model is trained on 840B tokens on Common Crawl, the size of the word vectors is 300.

[3] https://dl.fbaipublicfiles.com/fasttext/vectors-english/crawl-300d-2M-subword.zip

[4] https://nlp.stanford.edu/projects/glove/

The pre-trained word embedding can be downloaded at [5]. To preprocess the table as easily as fastText, we need to add one line to the table with the number of words and the dimension of the embedding on top of the GloVe table.

The command that we need is ’sed’. In practice we count the lines of the GloVe table, then we use:

```
$ wget http://nlp.stanford.edu/data/glove.840B.300d.zip
```

```
$ sed -i ’1s/^/2196017 300\n/’ glove.840B.300d.txt
```

The number 2196017 can be obtained by counting the lines that comprise the embedding file, while 300 is the dimension of the embedding.

4.2.2 Recurrent Neural Networks

Recurrent Neural Networks are a class of neural networks that is used to classify series of data. A series is defined as “a number of things or events of the same class coming one after another in spatial or temporal succession” [6]. The correlation between samples is what differentiates the series from a mere collection of samples.

RNNs are designed to capture such correlation via a hidden state, which keeps track of the context determined by the previous samples. This is different from the learning phase of

[5] http://nlp.stanford.edu/data/glove.840B.300d.zip

[6] https://www.merriam-webster.com/dictionary/series

the Feed-Forward Neural Networks, RNNs can differentiate their response to particular stimuli depending on the knowledge accumulated from previous events in the series.

This peculiarity defines a class of applications for which RNNs (and their extensions) are proved to be most effective: time-series prediction, speech recognition, machine translation, etc.

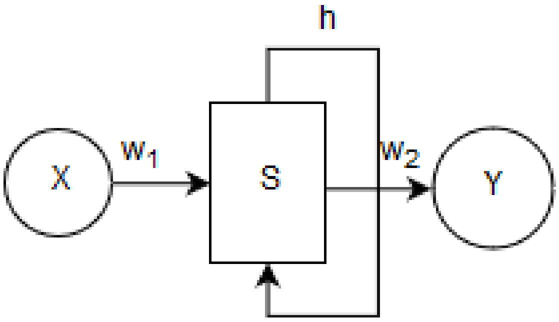


Figure 4. Diagram of a basic RNN cell.

$$S := \tanh(w_1 S + hX) \tag{4.17}$$

$$Y = w_2 S \tag{4.18}$$

While RNNs are considered a fairly good architecture for sequential data, they are particularly difficult to train because of several issues that could arise.

RNNs perform poorly when dealing with long term dependencies. If a piece of information is captured and it is needed after a very large gap, the information should be stored and reused at convenience. Unfortunately, RNNs usually do not learn how to successfully link long-term dependencies.

Recurrent Neural Networks suffer from the Vanishing Gradient Problem. Information travels in RNN from one input to the next in the series, then compute the error for each time step. Finally, back-propagate the cost function using gradient descent. This step is particularly crucial because the gradient is back-propagated all the way through all the RNN cells, updating sequentially each weight by the partial derivative of the error function with respect to the current weight. When we use traditional activation functions such as tanh the gradient output range is (0,1), therefore, back-propagation would only cause several chain multiplications by a small value, this decreases the variation applied to each weight in the network and slows down the training process, preventing the network to train properly.

On the other hand, if the gradients of the activation functions across the network were not limited to the interval (0,1), we could slip into the opposite problem. The Exploding Gradient Problem occurs when the chain multiplications produce progressively large gradients that grow uncontrollably as the number of RNN cells increases.

4.2.3 Long Short-Term Memory

Long Short-Term Memory cells are a variation of Recurrent Neural Networks that were introduced as a solution to the problems of Vanishing Gradient and Exploding Gradient. Their special characteristic is that they are capable of learning long-term dependencies, thanks to their gated internal structure. [Hochreiter and Schmidhuber, 1997].

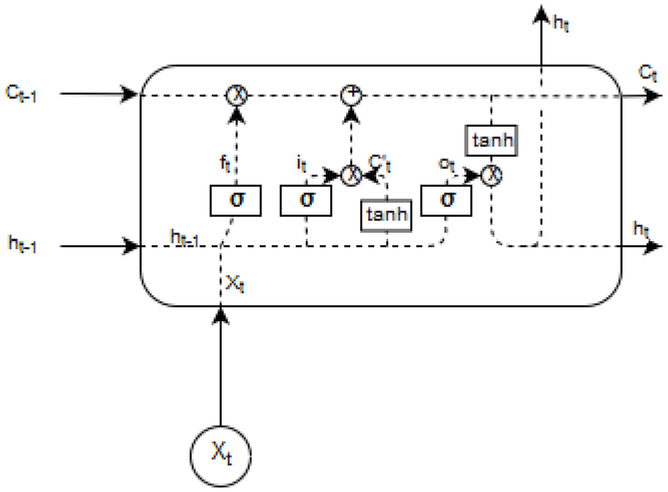


Figure 5. Long Short-Term Memory cell diagram.

To understand how LSTM cells work it may be useful to analyze separately the gated units that comprise the architecture. Let us define C_t the cell state a time t and h_t the hidden state at time t.

The forget gate defines if the information currently in the cell state has to be forgotten or is still - to some extent - useful for our purposes. σ corresponds to the sigmoid activation function, it outputs a number in range (0,1) corresponding to a scaling factor for the cell state at time C_{t-1}. If f_t is close to 1, it means that the information in C_{t-1} is still useful, if, instead, f_t is close to 0, it means that the previous state has to be forgotten.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{4.19}$$

The input gate determines, on the other hand, the contribution of the previous hidden state h_{t-1} and the current input x_t. The cell state is updated using tanh as activation function

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{4.20}$$

$$C'_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{4.21}$$

Now, we know how to perform the update on the cell state using outputs of the input and the forget gates.

$$C_t = f_t * C_{t-1} + i_t * C'_t \tag{4.22}$$

Now that C_t has been updated we have to output the hidden state and the cell state to the following LSTM cells.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{4.23}$$

$$h_t = o_t * \tanh(C_t) \tag{4.24}$$

In order to use LSTM cells for Natual Language Processing and Text Classification purposes we need to create a layer by stacking n LSTM cells in such a way that the output of the i^{th}

cell is fed as the input of the $(i+1)^{th}$. Together with the output of previous LSTMs, the cells require the text in the form of a sequence of n words.

The words in the text are vectorized and input in the LSTM cells. The resulting word embedding matrix $V \epsilon \mathbf{R}^{n+1,D}$ has n rows, one for each word in the sequence plus one final $(n+1)^{th}$ row with the embedding of the EOS (End of Sequence) token. D is the dimension of the embedded vectorized representation.

The resulting architecture is depicted in Figure Figure 6.

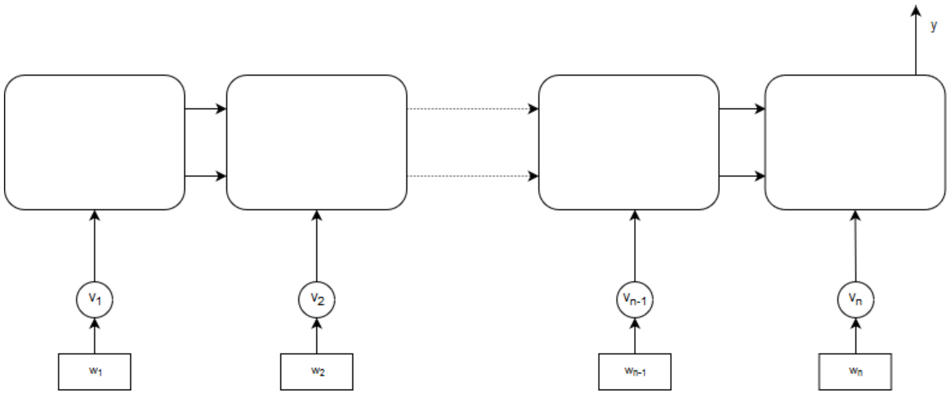


Figure 6. Series of LSTM cells for text classification.

4.2.4 Bidirectional Long Short-Term Memory

Bi-LSTMs are an extension to the LSTM architecture. Their main purpose is to capture the long-term dependencies between the current element and both the previous and the following sequence of words in the text. The LSTM architecture is modified by adding a further LSTM layer on the reversed input sequence. They are proved most useful when a deeper internalization of the context is needed. Thus, they perform successfully in works such as sentiment classification, emotion detection [Sharfuddin et al., 2018].

The layer of LSTM cells work on the natural order of words in the text, in accordance with Figure 6. In addition, we arrange a new set of LSTM cells that will operate on the backwards sequence.

The difference between CNN and Bi-LSTM is that "Bi-LSTM captures contextual semantics of a given word by means of its preceding and following information in the text, while CNN is used to capture structure information from the local contexts" [Burel et al., 2017].

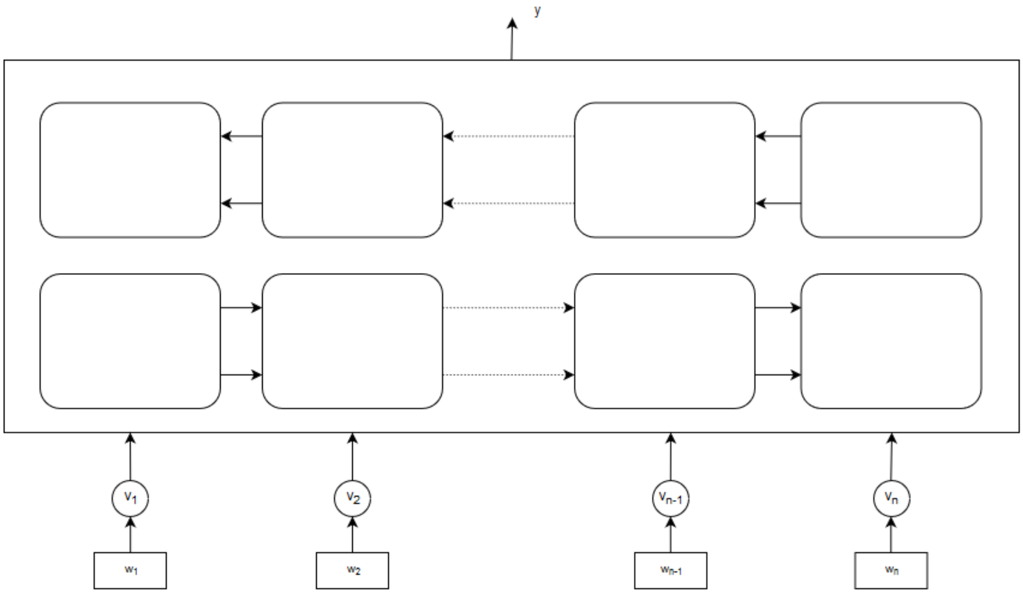


Figure 7. Bi-LSTM architecture for Text Classification.

Starting from the sequence of words in the document, we build a embedding matrix as specified for LSTMs. We use the vectorized representation of the i^{th} word as x input for both the layers of LSTM cells. The intra-layer connections are carried out by inputting the hidden state and the cell state into the current cell from the previous one, both in the normal sequence - first layer of LSTMs - and in the reversed sequence - second layer of LSTMs -.

Model Desgin

In our study we worked with the implementation from [7]. Firstly, we need to generate a vocabulary file with from the corpus. Let us define the directory where the vocabulary should be stored,’data_dir’, the input data directory ’input_dir’.

```
$ data_dir=tmp/rt

$ python gen_vocab.py --output_dir=$data_dir --dataset=rt --rt_input_dir=

    $input_dir --lowercase=False

$ wc -l tmp/rt/vocab.txt
```

Then, we save the size of the vocabulary in ’vocab_size’, we will need it in the training phases.

```
$ python gen_data.py --output_dir=$data_dir --dataset=rt --rt_input_dir=

    $inpu_dir$ --lowercase=False --label_gain=False
```

[7] https://github.com/tensorflow/models/tree/master/research/adversarial_text

Let us prepare for the training phase. Firstly, we define ’pretrain_dir’, the directory where the pretraining data is going to be stored.

```
$ python pretrain.py --train_dir=$pretrain_dir --data_dir=$data_dir --
    vocab_size=$vocab_size --embedding_dims=256 --rnn_cell_size=1024 --
    num_candidate_samples=1024 --batch_size=256 --learning_rate=0.001 --
    learning_rate_decay_factor=0.9999 --max_steps=1000 --max_grad_norm=1.0
    --num_timesteps=400 --keep_prob_emb=0.5 --normalize_embeddings
```

Let us define the embedding size, the size of the rnn cell, the batch size, the learning rate, its decay factor, the maximum number of steps and the related parameters.

At this point, the pretraining phase is complete and we define the training directory.

```
$ python train_classifier.py --train_dir=$train_dir --pretrained_model_dir=
    $pretrain_dir --data_dir=$data_dir --vocab_size=vocab_size --
    embedding_dims=256 --rnn_cell_size=1024 --cl_num_layers=1 cl_hidden_size
    =30 --batch_size=64 --learning_rate=0.0005 --learning_rate_decay_factor
    =0.9998 --max_steps=1500 --max_grad_norm=1.0 --num_timesteps=400 --
    keep_prob_emb=0.5 --normalize_embeddings --adv_training_method=al --
    perturb_norm_length=5.0
```

Finally, for the evaluation phase we need to define the evaluation directory and run the corresponding script.

```
$ python evaluate.py --eval_dir=$eval_dir --checkpoint_dir=$train_dir --
    eval_data=test --run_once --num_examples=1000 --data_dir=$data_dir --
    vocab_size=$vocab_size --embedding_dims=256 --rnn_cell_size=1024 --
    batch_size=256 --num_timesteps=400 --normalize_embeddings
```

4.3 Semi-Supervised Learning

Semi-supervised learning is a paradigm that combines labeled and unlabeled data to maximize the information gain, as compared to only using the former subset. The interest surrounding semi-supervised learning has grown in recent years; since the advent of social media, the magnitude of the data readily available has grown exponentially, albeit unlabeled in nature. On the other hand, labeled data is scarce, and producing reliable labeled data continues to be particularly expensive and time-consuming.

In the literature, numerous models try to exploit the presence of both labeled and unlabeled information, including self-training [Mihalcea, 2004], mixture models [Cozman et al., 2003], co-training [Blum and Mitchell, 1977], and graph-based methods [Subramanya and Talukdar, 2014].

Our semi-supervised learning section will focus on self-training applied to Naïve Bayes, Support-Vector Machines, and Convolutional Neural Networks. Consequently, we will use Virtual Adversarial Loss on Bidirectional Long Short-Term Memory models.

4.3.1 Self-training

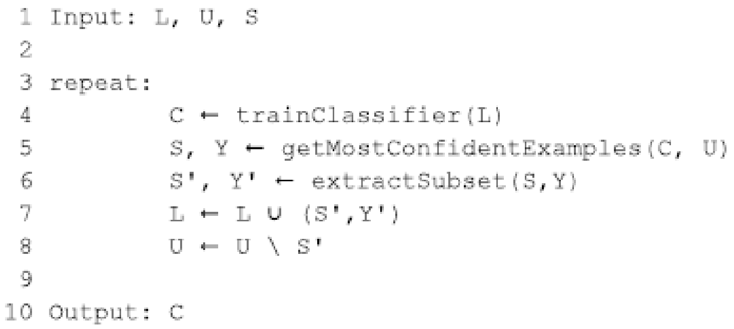
Self-training, also called self-teaching or bootstrapping, is a method that reintroduces high-confidence predictions of unlabeled inputs in the classifier. It is a so-called 'wrapper' method because the core-classifier is not changed by the algorithm [Kohavi and John, 1997]. Self-training 'wraps' around the model and introduces the most confident data in the labeled training set of the next iteration.

It is based on the assumptions that high-confidence predictions tend to be correct with well-separated classes [Zhu and Goldberg, 2009]. Nevertheless, if misclassifications occur, the data is associated with incorrect labels and reinforces the mistakes in the next iterations of the algorithm.

The main steps of our self-training implementation follow the algorithm by [Zhu and Goldberg, 2009] and are represented in Figure 8. Given the initial unlabeled and labeled data, U and L, respectively, we train the supervised classifier f. Secondly, we apply f to the unlabeled set. Finally, we add the above-threshold data, together with the predicted labels, to the labeled set, and, at the same time, we remove them from the unlabeled set.

Model Design

For our purposes, we added back into the labeled corpus the whole set of tweets whose prediction exceeded the confidence threshold. This helped us solving the problem of scarcity of unlabeled tweets and is shown to produce better results in semi-supervised learning for topic classification [Gollapalli et al., 2013], [Caragea et al., 2015]. Furthermore, we introduced a different terminal condition to account for tweets that were consistently below the confidence threshold. Our

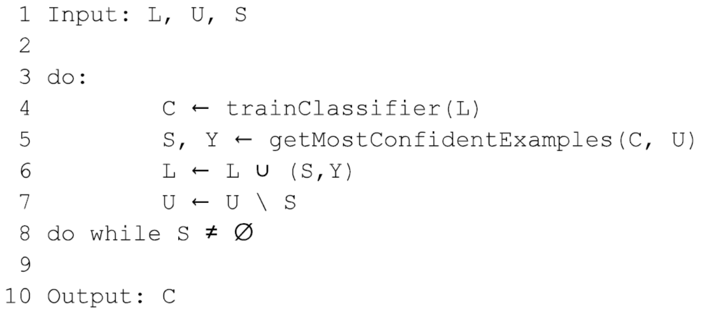


```
Input: L, U, S

repeat:
        C ← trainClassifier(L)
        S, Y ← getMostConfidentExamples(C, U)
        S', Y' ← extractSubset(S,Y)
        L ← L ∪ (S',Y')
        U ← U \ S'

Output: C
```

Figure 8. Self-training algorithm.

algorithm stops when two subsequent iterations share the same unlabeled set. In detail, instead of terminating when all the unlabeled messages are inserted back in the labeled training set ($U \neq \varnothing$), we terminate when the classifier cannot add any more messages to the set of confident messages ($S \neq \varnothing$). In the former case, the algorithm could remain stuck on a local optimum, when at least one data point is consistently below the threshold. Our choice attempts to solve this problem. Figure 9 represents our algorithm. L, U, S represent, respectively, the labeled set, the unlabeled set, and the holder for above-threshold data.

The following experiments with semi-supervised learning are carried out using a confidence threshold of 0.99. This value has been chosen as a middle ground between 0.9 and 0.999. The runs with 0.9 as confidence threshold are characterized by a large amount of data added to the training set in the first iterations of the algorithm. Being the confidence threshold is too low, a lot of unlabeled tweets that are added to the training set are classified erroneously, as we can



```
Input: L, U, S

do:
        C ← trainClassifier(L)
        S, Y ← getMostConfidentExamples(C, U)
        L ← L ∪ (S,Y)
        U ← U \ S
do while S ≠ ∅

Output: C
```

Figure 9. Our implementation of the self-training algorithm.

see from the confusion matrix of the unlabeled set. As discussed in section 5.1.3, the unlabeled set has been constructed from a portion of the labeled set so that the labels would still be stored and used to validate this analysis. Thus, it is entailed that misclassified unlabeled data is re-inserted in the training set and behaves as noise in the labeled set. The runs with 0.999 as confidence threshold add, conversely, a small amount of data at every iteration and tend not to differ from the supervised settings.

4.3.2 Virtual Adversarial Loss

Adversarial and Virtual Adversarial Loss functions are regularization methods for classifiers to improve robustness to perturbations.[Miyato et al., 2016] Adversarial Training and Virtual Adversarial Training differ for the learning tasks they serve. The former predicts labels while trying to improve the robustness against noisy perturbations. The latter "defines the adversarial direction without label information and is hence applicable to semi-supervised learning" [Miyato et al., 2018].

From [Goodfellow et al., 2014], [Miyato et al., 2018] Adversarial loss can be written as:

$$L_{adv}(x_l, \theta) = D[q(y|x_l), p(y|x_l + r_{adv}, \theta)] \tag{4.25}$$

where:

$$r_{adv} = \underset{r;||r|| \leq \epsilon}{\operatorname{argmax}} D[q(y|x_l), p(y, x_l + r, \theta)] \tag{4.26}$$

Virtual Adversarial Loss is given by:

$$L_{vadv}(x_*, \theta) = D[q(\cdot|x, \hat{\theta}), p(\cdot|x + r_{vadv}, \theta)] \tag{4.27}$$

$$r_{vadv} = \underset{r;||r|| \leq \epsilon}{\operatorname{argmax}} D[q(\cdot|x, \hat{\theta}), p(\cdot, x + r, \hat{\theta})] \tag{4.28}$$

D denotes the divergence between p and q. In both cases the classifier must account for the perturbations of the current x in the most sensitive direction. What changes is the label requirement: Adversarial Training tries to predict the label given the perturbed input, Virtual Adversarial Training does not require the label y and can be hence used for semi-supervised learning models.

[Miyato et al., 2016] proposes to apply this technique for text classification, the approach allows to add fixed norm perturbation on the embedded words and aims at calculating the effect of the virtual adversarial perturbation.

Model Design

In the semi-supervised setting, we need an unlabeled file to take part in the training phase. It is references as amazon_unlabeled_input_file and is parameterized with the set of unlabeled tweets.

The second modification regards the loss function, it is now possible to use 'Virtual Adversarial Loss' function as our error function. The parameter to modify, in this case, is adv_training_method and the value is 'val' which stands, in fact, for Virtual Adversarial Loss.

CHAPTER 5

EXPERIMENTAL DESIGN

5.1 Dataset

5.1.1 Retrieval

For the purpose of this analysis we collected data from two different sources.

The first dataset is publicly available with the name CrisiLexT26. It comprises a set of about 13'000 tweets referring to 26 events occurred in the years 2012 and 2013, the majority of which are in English but there is a non negligible percentage of tweets in other languages (Spanish, Italian, etc.). [1].

The collection is manually labeled by Figure Eight, also known as CrowdFlower, according to informativeness and relatedness to the disaster. [Olteanu et al., 2015]

The second is part of a Damage Assessment Study on 4 disasters that took place in Italy between 2009 and 2014. This is a set of about 5000 Italian tweets. [2]

The dataset has been annotated according to damage assessment, relevance and relatedness to the disaster.

[1] `https://github.com/sajao/CrisisLex/tree/master/data/CrisisLexT26/`

[2] `http://socialsensing.it/en/datasets`

5.1.2 Preprocessing

The data is loaded and processed using standard regular expressions, and punctuation marks from the Python string standard library. We removed all non-English characters, special characters, parentheses, space punctuation marks and converted the obtained string to lowercase.

Secondly, to account only for relevant tweets, we removed the set flagged as not relevant or not related to the disaster. Alternatively, one could think of a non-binary classification task with the classes 'informative', 'non-informative', and 'non-related / non-relevant'. However, for the subsequent experiments, we preferred to use binary classification with 'informative' class as positive and the 'non-informative' class as negative.

Finally, the data is collected and fed to the core algorithm, about 11500 tweets comprised the first dataset, and 4500 the second.

5.1.3 Partitioning

To assess the quality of the supervised learning and the semi-supervised learning approaches with the available data, we decided to split the datasets into two parts, the first part acts as the core of the labeled set, it is further divided into:

- Training set: the set of labeled tweets upon which the supervised classifier is built.
- Development set: labeled tweets that give a glimpse of how good the classifier is. A portion of the labeled set is randomly selected as the development set.
- Evaluation set: a set of tweets is randomly selected to be part of the test set. This data is preprocessed and saved in a directory 'eval_dir'. The content of the 'eval_dir' has to

be changed in order to test the classifier on custom data. The format of the data has to conform to the structure of the preprocessed datasets.

The second portion of the data emulates a set of unlabeled tweets. The labels are put aside and used only in the evaluation phase. Using the latest model, a label is assigned to each tweet and those above the threshold are added to the training set. In this step, the aforementioned 'stripped' labels allow building statistics about the added tweets.

In the next section we will present a more detailed explanation of how the corpora are partitioned and analyzed, unitedly.

5.2 Experimental Settings

The following describes how the experiments are organized and carried out. A large number of experiments has been carried out for this purpose, ∼50 for the English dataset and ∼60 for the Italian corpus. It is interesting that even though the datasets are analyzed using the same settings, the results will have some differences. This analysis will be presented in section 6.

5.2.1 Experiments

The following experiments are conducted according to the representation in Figure 10. The dataset is divided into two sections. The first, about 1/3 of the total data, contains the labeled tweets used as the training, development, and test sets. The second part is handled as a set of unlabeled tweets, the labels removed and used solely to evaluate the semi-supervised learning runs. These labels were not used to train the model.

The first set is divided into a training set, development set, and evaluation set. 1/10 of the labeled set is used as dev set. About 500 tweets composed the evaluation set; for clarity

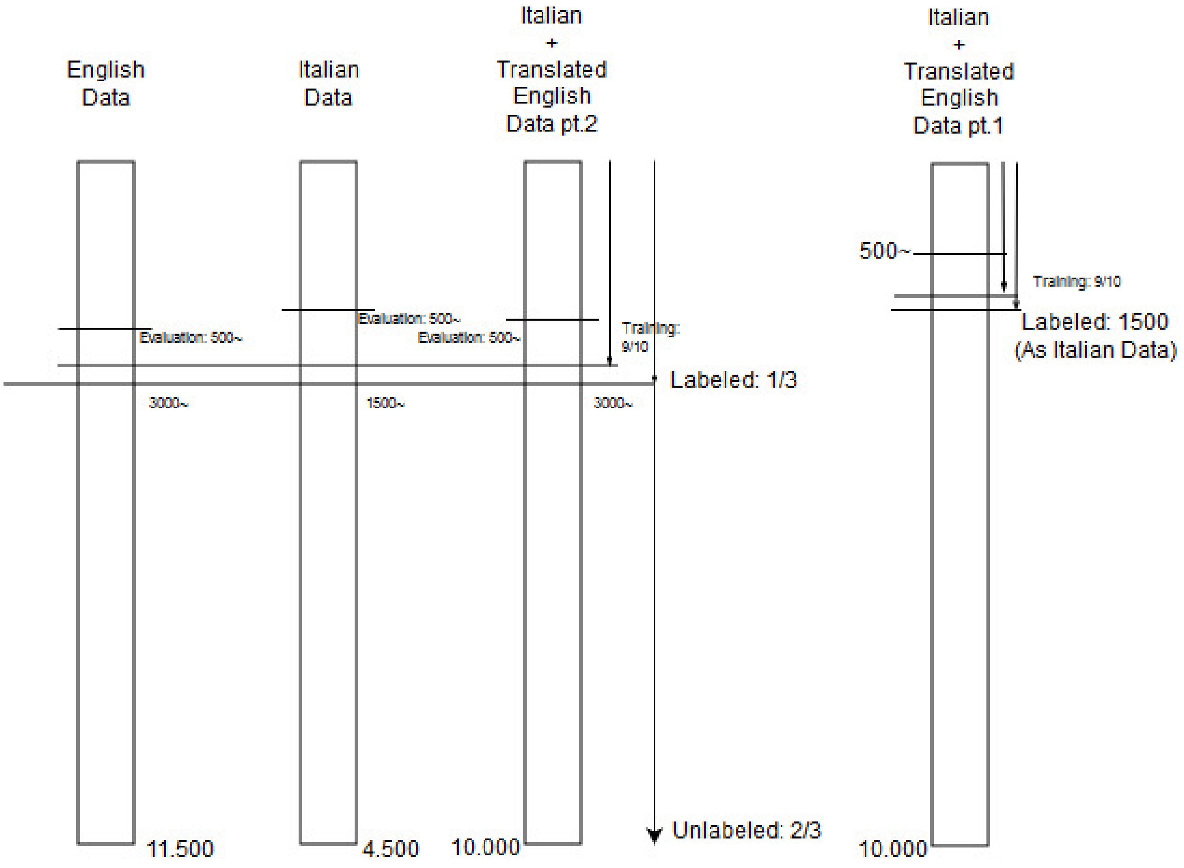


Figure 10. Database split across the experiments.

reasons, we decided to save this set in an evaluation directory. In this way, if we were to conduct evaluations on a customized set, we would only have to customize the evaluation directory.

From this starting point, the experiments are characterized by progressively smaller amounts of training data: 100%, 70%, 50%, 20%, and 10% were the standard percentages, however, we did not report some of the runs. There are two reasons for this occurrence: either the code could not be executed on lower percentages because of limits in the architecture, this is the case of Bi-LSTMs on the Italian dataset and 10% of the English dataset, or the classifier in the previous instance was so poor (all the data points assigned to one of the classes, for example) that training on even smaller sets would be worthless.

The architectures used to train the classifiers are of various types, the runs are executed on Naïve Bayes, Support-Vector Machines trained with Stochastic Gradient Descent, Convolutional Neural Networks with Word2Vec embedding by gensim, with GloVe embedding, with fastText embedding, Bi-LSTM trained with (Virtual) Adversarial Loss function. In section 6 we will analyze in detail the features of these architectures and the outcomes of the runs.

Additional Experiments on the Italian Corpus

The last part of section 6 is devoted to the analysis of methods that try to compensate for the scarcity of Italian data.

Starting from the most successful English run with CNN (English 100% + fastText) the most confident unlabeled tweets are translated into Italian via the Google Translate API.

Thus, we set up two additional experiments. In the first, we simply add the translated tweets to the unlabeled set, keeping the labeled portion as-is (1500 tweets comprised the labeled set). In the second run, we add the unlabeled set, together with the predicted labels, as part of the labeled set, then we partition the data according to the subdivision in sections 5.1.3 (3000 tweets comprised the labeled set).

CHAPTER 6

RESULTS AND ANALYSIS

6.1 Results

6.1.1 Supervised Learning

In Table I we show that English CNNs reach an accuracy of 0.84 when using fastText, 0.83 using GloVe and 0.81 when using Word2Vec, while using Bi-LSTMs the accuracy rockets to 0.94. When using Machine Learning techniques the accuracy is 0.87 (SVM) and 0.79 (NB). The accuracy with SVM is not indicative of a better performance than CNN since the f1-measure is around 0.88 with fastText and GloVe but it plummets to 0.79 when using SVM (Figure 12). The pattern replicates with smaller percentages but, the performance slowly degrades for smaller percentages of the training data, as expected.

Considering the data in table Table I, fastText performs generally better than GloVe; these two embeddings outperform Word2Vec. It is interesting to notice that Word2Vec and SVM perform better on lower percentages than fastText and GloVe. Even more surprisingly, Bi-LSTM performance (Table II) degradation is close to zero: the accuracy with 100% of the training data is similar to the accuracy with 20%. Unfortunately, it was impossible to work with lower percentages in the case of Bi-LSTM because of architectural limits. English Bi-LSTM peaks at 0.954 in accuracy; in our knowledge, this value outperforms the current research standard on Deep Learning models.

Let us analyze the case of the Italian language in Table III. CNN instances perform accurately 0.941, 0.91, 0.90 with fastText, GloVe and Word2Vec as embedding layers, respectively. Machine Learning techniques perform well, SVM and Naïve Bayes reach 0.94 and 0.90, respectively. SVMs seem not to suffer from performance degradation in the largest runs (100%, 70% and 50% of the training data). Unfortunately, it has not been possible to evaluate Bi-LSTMs in the supervised learning phase because of the small amount of Italian data available.

6.1.2 Semi-Supervised Learning

In the English case the performance of Bi-LSTMs is higher than any other Machine Learning and CNN instances, we have, on average, ∼0.95 accuracy, while the closer CNN gets to this value is 0.90 with fastText. Pivoting from the supervised learning paradigm to the semi-supervised learning case, Bi-LSTM performance improves in the case of 70% and 50% of the labeled data from 0.952 and 0.954 to 0.961 and 0.959, respectively, and remains similar for 100% and 20%, going from 0.944 to 0.942 in the former case and from 0.944 to 0.943 in the latter.

The Semi-supervised setting continues to have good performance in many of the other settings, in the case of fastText and 100% of the labeled data, the f1-measure soars from 0.88 to 0.93, when using GloVe 100% the f1-measure goes from 0.88 to 0.91. With Machine Learning instances, low metrics could be due to the non-English tweets in CrisisLexT26, in which case an English tokenization could deteriorate the performance.

In Figure 11 the accuracy score is represented on the y axis, whereas the percentage of the training data is shown on the x axis. Bi-LSTMs steadily outperform other Machine Learning and Deep Learning algorithms. The semi-supervised settings perform slightly better than the supervised runs. Furthermore, CNNs perform fairly well, with the semi-supervised setting outperforming the supervised instance, and peak in accuracy with a value of 0.89. For the sake of clarity, NB classifiers, SVMs, CNN with Word2Vec are not depicted in Figure 11, since we do not report valuable performance by any of these architectures.

On the whole, Figure 11 shows that Bi-LSTM runs obtain up to ∼96% in classification accuracy, semi-supervised approaches with CNN(fastText) and CNN(GloVe) follow with ∼89% and ∼87%.

The Bi-LSTM runs are very successful, nevertheless it may be useful to characterize the behavior of CNNs and SVMs. In exploring the f1-measure for CNNs and Machine Learning models, Figure 12 highlights an interestingly similar pattern. Performance in the semi-supervised CNN runs improves when compared to the corresponding supervised instances. For instance, when using 100% of the training data, our f1-measure on CNN with FastText improves by a 4.4% factor (Figure 13), with GloVe, it grows by 2.3% and with Word2Vec by 1.3%.

In the Italian case the performance of Bi-LSTM (semi-supervised) is also higher than any other technique used, in fact we reach 0.969. The second best is obtained by CNN with fastText (supervised), with a value of 0.941. In general, several semi-supervised Machine Learning and Deep Learning models do not have satisfactory improvement over the supervised runs. The only cases in which the semi-supervised task perform better are CNN(Word2Vec) and SVM. One of the reasons could be that the amount of labeled data is sufficient enough to build a strong classifier, but not sufficient to account for variation of the data. A in-depth analysis of the results has been carried out in 6.2

Finally, Figure 14 shows the performance on the Italian Corpus. Bi-LSTM is confirmed the most accurate among the models. Differently from the previous scenarios, the CNNs with fastText and GloVe do not achieve a significant improvement compared to SVM and NB classifiers. Furthermore, CNNs with fastText and GloVe slightly decrease in performance when a semi-supervised setting is adopted.

6.2 Analysis

As regards both the English and the Italian datasets it is noticeable that the Bi-LSTMs perform better than the CNN with any embedding, and the CNN has better results than Machine Learning methods (Support-Vector Machines + Stochastic Gradient Descent and Multinomial Naïve Bayes). This is deemed to conform to the complexity of these algorithms and is supported by their general accomplishments on text mining applications. Support-Vector machines obtain very good results with the Italian dataset in both supervised and semi-supervised learning instances.

One of the differences that could be impactful when analyzing the results is that the CrisisLexT26 dataset contains some tweets in Spanish, Italian, Portuguese and the language difference has a negative impact on some architectures that we employed. Namely, SVM and NB use an English tokenizer, whereas CNN instances with keyed vectors like fastText and GloVe use English embeddings. In the former case, with Machine Learning techniques, the tokenization could mistakenly modify some of the text into useless tokens. In the latter, with CNNs, the English embedding would not account for (some of the) non-English words.

When analyzing the results of the semi-supervised runs in both English and Italian, several factors have to be considered.

Firstly, a recurrent effect of training the classifier on smaller percentages is that the model is not strong enough to achieve good results in the classification task, therefore, it cannot classify properly the unlabeled set. This may result in performance deterioration. Researchers have shown that training on a particularly small set of labels could have a negative impact on the classifier [Zhang and Vucetic, 2016]. This phenomenon becomes evident in the Italian dataset when working on CNN and NB with 10% of the labeled set and in English when using SVM and CNN(GloVe) on <20% and fastText on 10% of the data.

Let us take English + CNN(GloVe) 20% as an example. In this case precision plunges from 0.73 to 0.67, but recall jumps from 0.93 to 0.99, this means that the positive class was predominant and many tweets were classified as positive, even if they belonged to the negative class.

One more reason why this aspect would be impactful is that the distribution of classes is unbalanced in both datasets. This could become a problem with particularly small percentages of training data, because, being the classifier too poor, it could internalize the label unbalance and the skew in the data in the model.

Overall, SVMs seem to outperform CNNs in the majority of the Italian experiments, this confirms that Support-Vector Machines may outperform other techniques in situations with inadequate training data [Manning et al., 2008].

In regards with a broader analysis of the data, English and Italian runs have some interesting differences that is worth pointing out. Italian and English Semi-supervised instances perform well with larger percentages of training data. In the Italian case the Bi-LSTM instance is confirmed to perform better than all others architectures. Nonetheless, the training dataset is not large enough to run the algorithm for the supervised setting and for percentages smaller than 100% in the semi-supervised setting.

The GloVe embedding required further analysis. The vectors are supposedly created for the English language, but, after a successful training attempt on the Italian dataset, we decided to further inspect the correspondences between the Italian text and the entries in the GloVe embedding table. We counted how many Italian words were found in the embedding table, and calculated their percentage over all the words in the Italian dataset. The results on the Italian corpus had the same order of magnitude as the corresponding results on the English corpus, therefore we decided to keep GloVe as part of the experiments on the Italian corpus. However, from Table I and Table III we notice that GloVe has a poorer behavior in Italian, in fact it performs slightly better than Word2Vec. Given the considerations about the nature of GloVe, we consider this behavior as aligned with the predictions.

All things considered, the following must be taken into account when working with non-English corpora. Although some words have the same spelling both in Italian and in English, they may take on different meanings: ’ride’, ’pace’, ’agenda’, ’bimbo’, ’brave’, ’cave’, ’date’, ’figure’, ’positive’, ’salute’ etc. From the assumption of GloVe as an English embedding, we derived that those words would still have a vectorized representation (because they are part of the English dictionary). However, the vector would correspond to a word with the same syntactic representation and different semantics. This has a small, yet non-negligible, impact on the Italian runs with CNN embedded with GloVe.

Additional Analysis on the Italian Corpus

In the case of the Italian tweets, self-training has not obtained satisfactory results with (semi-supervised) CNNs. In attempt to look for the reason behind this phenomenon, we tried to think of the main differences between the English and the Italian dataset. The first difference that comes to mind is that the two corpora have different magnitudes, CrisisLexT26 contains 13000 tweets and Cresci contains about 4500 tweets, about 1/3 of the data. Researchers have pointed out that not having a large unlabeled corpus could be detrimental to the performance [Zhang and Vucetic, 2016]. Thus, in order to bridge this gap, we included the set of confident unlabeled tweets in the labeled set as described in the sections 5.1.3 and 5.2.1.

The first of the two approaches consisted in keeping the labeled set unvaried (1500 tweets) and adding the tweets as part of the unlabeled set. This experiment is illustrated in Table V and the performance is similar to the semi-supervised case. In the second experiment we increased the number of training tweets. This brings a gain on the overall performance, but comparing this data to the English experiments proves that the improvement on the English instances remains unmatched. Let us now analyze this behaviour.

Firstly, above-threshold tweets do not prove that their classification is correct, they are still added to the labeled set but they could be reinforcing false assumptions on the data [Zhu and

Goldberg, 2009]. A similar, yet less erroneous approach would be to translate the labeled set, associate it with the correct label and eventually insert this data in the Italian labeled set.

Secondly, translating text causes issues of its own: accuracy of the translation and correct phrasing are often difficult to evaluate and it becomes an even more challenging task when dealing with brief text. The first classification task does not result in any kind of improvement with regard to the corresponding semi-supervised instances. It seems that, in this case, the translated tweets cause a noisier unlabeled set, and the model is not robust enough to discern noiseless data and add it to the training set. The second experiment brings the semi-supervised performance closer to the supervised setting (outperforming it, in some instances) which is an improvement when considering the previous task. However, we consider it partially unsuccessful when comparing the results with the English data. Nevertheless, the classifier appears to be less prone to errors and more robust against noise, which can still be considered as an positive achievement.

TABLE I: English CNN.

	Supervised Learning	Semi-Supervised Learning	Supervised Learning	Semi-Supervised Learning	Supervised Learning	Semi-Supervised Learning
	CNN + fastText	CNN + fastText	CNN + GloVe	CNN + GloVe	CNN + Word2Vec	CNN + Word2Vec
1.000						
accuracy	0.837	0.898	0.830	0.864	0.810	0.824
precision	0.875	0.921	0.873	0.895	0.844	0.836
recall	0.889	0.930	0.890	0.915	0.886	0.925
f-measure	0.882	0.926	0.882	0.905	0.865	0.878
0.700						
accuracy	0.822	0.851	0.806	0.828	0.808	0.808
precision	0.860	0.866	0.850	0.823	0.828	0.813
recall	0.869	0.914	0.847	0.932	0.895	0.922
f-measure	0.865	0.889	0.849	0.874	0.860	0.864
0.500						
accuracy	0.743	0.793	0.782	0.838	0.799	0.797
precision	0.759	0.796	0.860	0.834	0.797	0.824
recall	0.866	0.929	0.816	0.956	0.925	0.871
f-measure	0.809	0.858	0.838	0.891	0.856	0.847
0.200						
accuracy	0.729	0.790	0.737	0.685	0.763	0.743
precision	0.746	0.789	0.733	0.673	0.834	0.819
recall	0.883	0.923	0.928	0.990	0.800	0.783
f-measure	0.809	0.851	0.819	0.801	0.817	0.801
0.100						
accuracy	0.700	0.673			0.770	0.759
precision	0.710	0.669			0.810	0.824
recall	0.923	1.000			0.855	0.811
f-measure	0.802	0.802			0.832	0.817

TABLE II: English SVM, NB, Bi-LSTM

	Supervised Learning	Semi-Supervised Learning	Supervised Learning	Semi-Supervised Learning	Supervised Learning	Semi-Supervised Learning
	SVM	SVM	NB	NB	Bi-LSTM	Bi-LSTM
1.000						
accuracy	0.873	0.866	0.793	0.683	0.944	0.942
precision	0.864	0.874	0.877	0.750		
recall	0.723	0.688	0.464	0.118		
f-measure	0.788	0.770	0.607	0.203		
0.700						
accuracy	0.829	0.820	0.784	0.679	0.952	0.961
precision	0.767	0.845	0.862	0.000		
recall	0.680	0.547	0.389	0.000		
f-measure	0.721	0.664	0.536	0.000		
0.500						
accuracy	0.821	0.785			0.954	0.959
precision	0.769	0.790				
recall	0.634	0.451				
f-measure	0.695	0.574				
0.200						
accuracy	0.776	0.743			0.944	0.943
precision	0.667	0.833				
recall	0.557	0.214				
f-measure	0.607	0.341				
0.100						
accuracy	0.727	0.705				
precision	0.612	0.731				
recall	0.414	0.131				
f-measure	0.494	0.222				

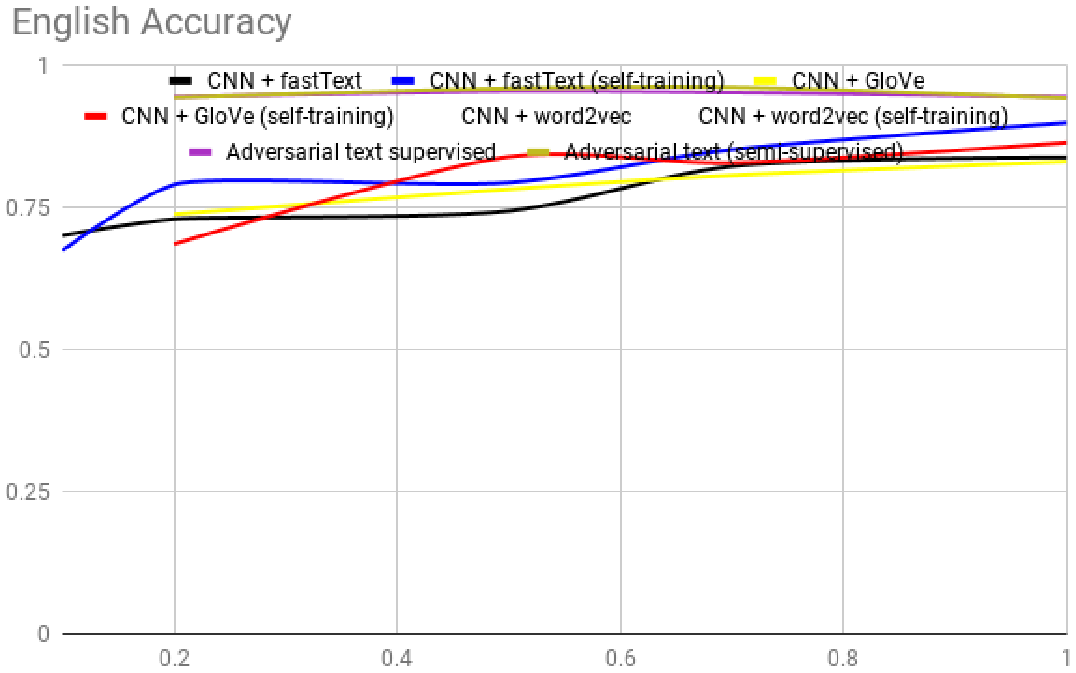


Figure 11. Accuracy on binary classification: English dataset.

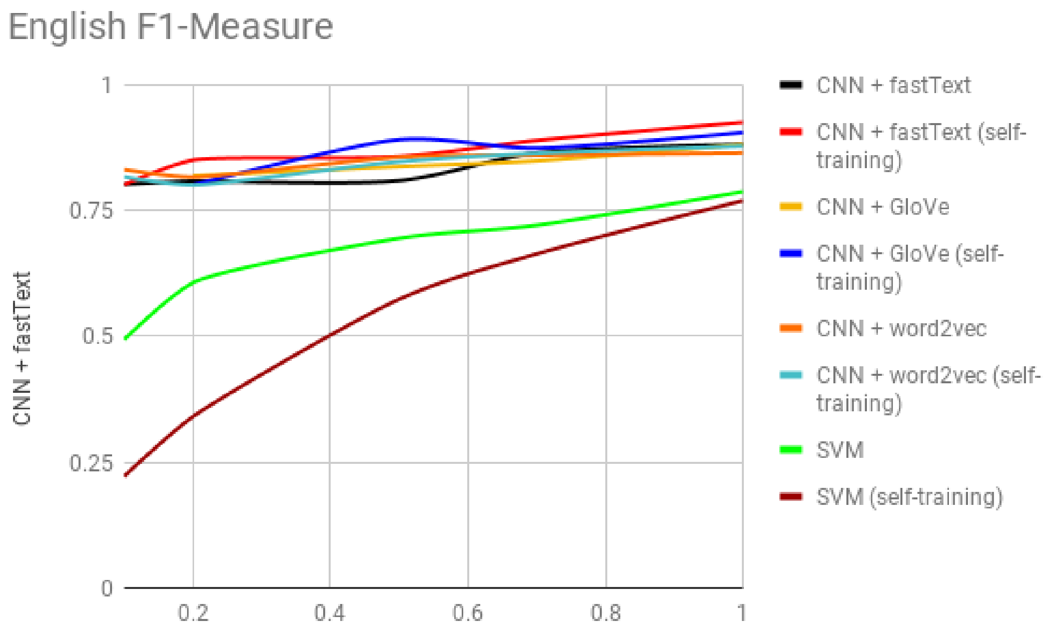


Figure 12. F1-measure on CNNs and SVMs: English dataset.

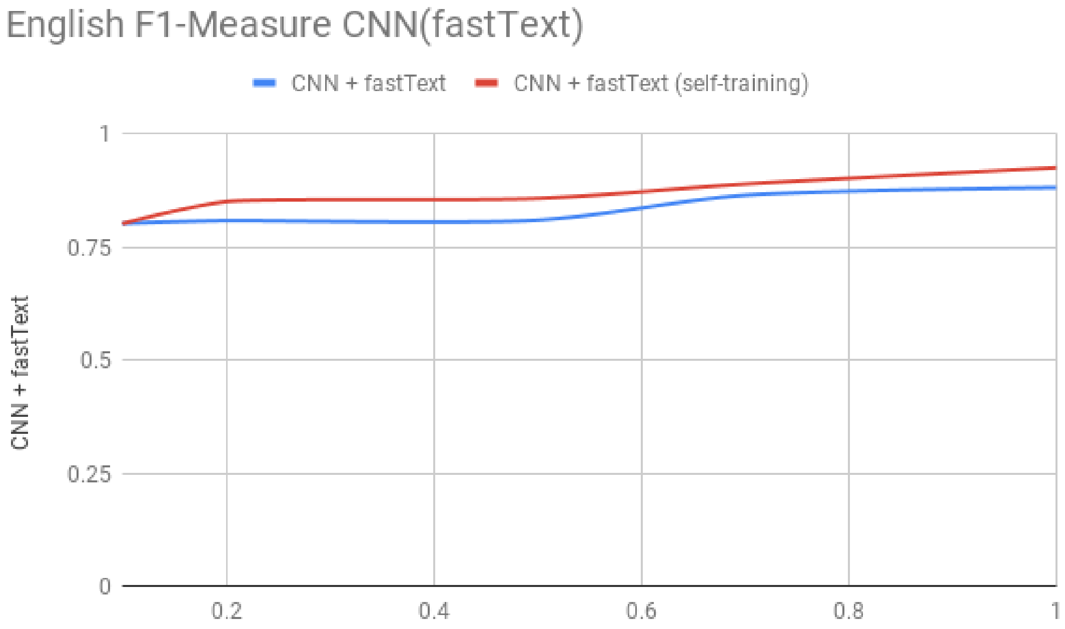


Figure 13. Performance on binary classification on English dataset.

TABLE III: Italian CNN.

	Supervised Learning	Semi-Supervised Learning	Supervised Learning	Semi-Supervised Learning	Supervised Learning	Semi-Supervised Learning
	CNN + fastText	CNN + fastText	CNN + GloVe	CNN + GloVe	CNN + Word2Vec	CNN + Word2Vec
1.000						
accuracy	0.941	0.915	0.910	0.894	0.901	0.893
precision	0.850	0.796	0.883	0.830	0.789	0.772
recall	0.947	0.917	0.796	0.803	0.853	0.846
f-measure	0.896	0.852	0.837	0.816	0.820	0.807
0.700						
accuracy	0.933	0.917	0.897	0.887	0.878	0.895
precision	0.871	0.821	0.825	0.807	0.770	0.809
recall	0.878	0.878	0.846	0.833	0.775	0.797
f-measure	0.875	0.849	0.835	0.820	0.773	0.803
0.500						
accuracy	0.920	0.918	0.908	0.885	0.889	0.905
precision	0.830	0.838	0.892	0.876	0.863	0.912
recall	0.884	0.862	0.790	0.720	0.748	0.755
f-measure	0.856	0.850	0.838	0.790	0.801	0.826
0.200						
accuracy	0.886	0.863	0.888	0.855	0.869	0.867
precision	0.868	0.875	0.784	0.686	0.769	0.741
recall	0.719	0.623	0.813	0.866	0.730	0.774
f-measure	0.787	0.728	0.799	0.766	0.749	0.757
0.100						
accuracy	0.879	0.845	0.837	0.758	0.812	0.802
precision	0.724	0.909	0.766	0.840	0.775	0.759
recall	0.778	0.370	0.686	0.275	0.548	0.522
f-measure	0.750	0.526	0.724	0.475	0.642	0.619

TABLE IV: Italian SVM, NB, Bi-LSTM.

	Supervised Learning	Semi-Supervised Learning	Supervised Learning	Semi-Supervised Learning	Supervised Learning	Semi-Supervised Learning
	SVM	SVM	NB	NB	Bi-LSTM	Bi-LSTM
1.000						
accuracy	0.937	0.935	0.900	0.893		0.969
precision	0.953	0.946	0.895	0.882		
recall	0.956	0.962	0.973	0.979		
f-measure	0.955	0.954	0.932	0.928		
0.700						
accuracy	0.941	0.943	0.873	0.851		
precision	0.959	0.965	0.873	0.849		
recall	0.957	0.954	0.963	0.963		
f-measure	0.958	0.959	0.916	0.903		
0.500						
accuracy	0.916	0.918	0.873	0.846		
precision	0.931	0.923	0.873	0.837		
recall	0.948	0.960	0.968	0.980		
f-measure	0.939	0.941	0.918	0.903		
0.200						
accuracy	0.888	0.892	0.846	0.697		
precision	0.939	0.920	0.837	0.697		
recall	0.902	0.930	0.992	1.000		
f-measure	0.920	0.925	0.866	0.821		
0.100						
accuracy	0.895	0.905				
precision	0.923	0.920				
recall	0.937	0.957				
f-measure	0.930	0.938				

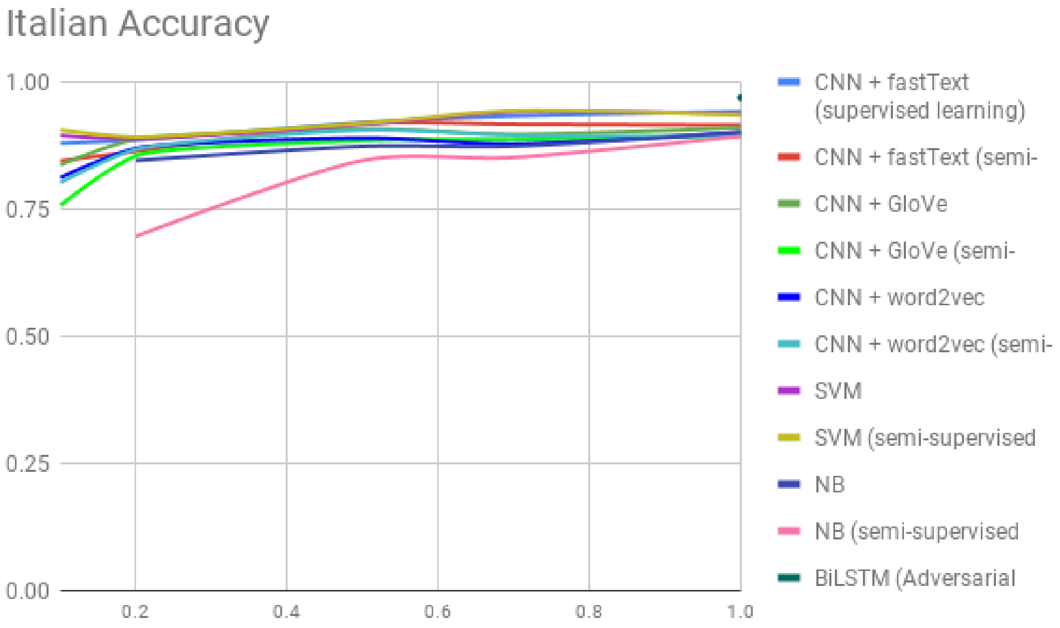


Figure 14. Accuracy on binary classification: Italian dataset.

TABLE V: Italian CNN with additional translated tweets.

	Supervised Learning	Semi-Supervised Learning	Supervised Learning	Semi-Supervised Learning
	CNN + fastText	CNN + fastText	CNN + fastText	CNN + fastText
1.000	1500 Labeled		3000 Labeled	
accuracy	0.921	0.911	0.936	0.936
precision	0.878	0.853	0.862	0.855
recall	0.860	0.853	0.899	0.911
f-measure	0.869	0.853	0.880	0.882
0.700				
accuracy	0.928	0.893	0.917	0.912
precision	0.891	0.796	0.857	0.832
recall	0.848	0.834	0.842	0.852
f-measure	0.869	0.815	0.849	0.842
0.500				
accuracy	0.911	0.879	0.926	0.909
precision	0.832	0.792	0.879	0.828
recall	0.876	0.797	0.850	0.846
f-measure	0.854	0.795	0.864	0.837
0.200				
accuracy	0.875	0.845	0.911	0.909
precision	0.885	0.901	0.854	0.807
recall	0.639	0.507	0.810	0.872
f-measure	0.742	0.649	0.831	0.839
0.100				
accuracy	0.875	0.848	0.893	0.875
precision	0.818	0.840	0.836	0.768
recall	0.723	0.577	0.701	0.762
f-measure	0.767	0.684	0.762	0.765

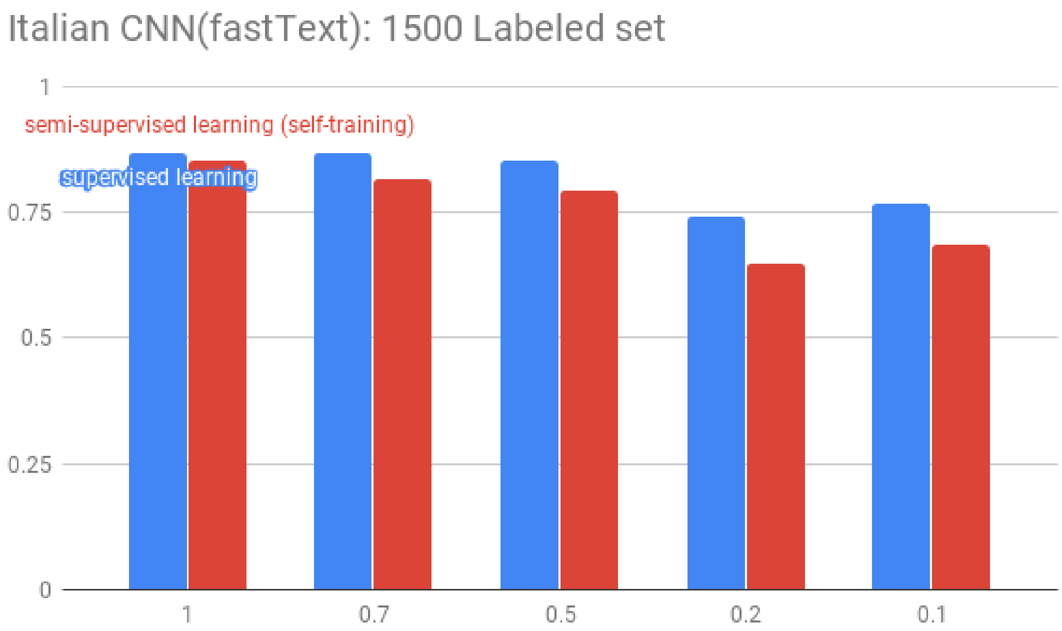


Figure 15. F1-measure on binary classification: Italian dataset & translated tweets (1500 Labeled).

6.3 Comparison with Related Work

This section aims at comparing our results with other state of the art papers. Given the heterogeneity and the extent of the contributions in this field, our work will not focus on exhaustiveness, but will rather point out the strong points of our research, when compared to previous work. [Olteanu et al., 2015], [Alam et al., 2019], [Imran et al., 2014], [Neppalli et al., 2018].

Our F1-measure with CNN + fastText reaches 0.882 in the supervised setting, this result is comparable to 0.885, weighted average of F1-measure by [Neppalli et al., 2018]. Differently

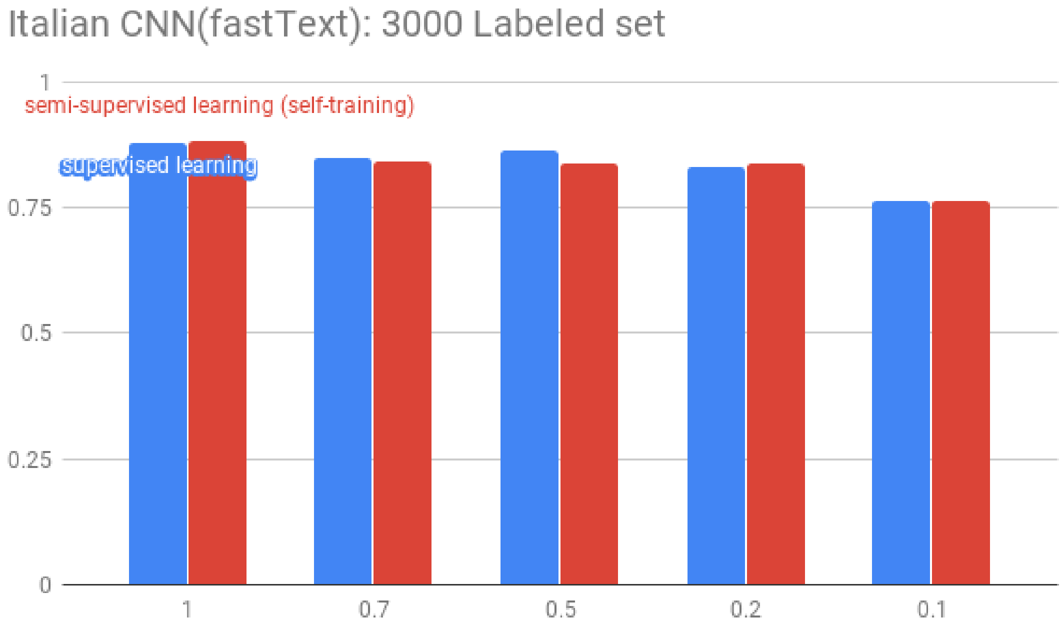


Figure 16. F1-measure on binary classification: Italian dataset & translated tweets (3000 Labeled).

from their work, we do not remove non-English tweets, we do not employ cross-valiation, and our supervised learning run uses, in preparation for the semi-supervised setting, less than 1/3 of the labeled set, leaving out the unlabeled portion as described in Figure 10. Our F1-measure soars to 0.926 in the self-training instance. We also obtain similar results in the Naïve Bayes classifier, when analyzing the supervised setting. In this case, our semi-supervised setting does not improve the performance of the classifier.

[Alam et al., 2019] constructs a similar analysis, their F1 measure reaches 0.93 for informativeness. Our semi-supervised run matches approximately this result, 0.926. Interestingly, this result has been achieved using only on third of the labeled set, the remaining has been stripped of the labels and considered as the unlabeled set.

As regards more sophisticated techniques, our Bi-LSTM with Virtual Adversarial Loss classifies informativeness with an accuracy ranging from 0.94 to 0.95 in the supervised runs, from 0.94 to 0.96 in the semi-supervised runs, against 0.93 with the CNN by [Alam et al., 2019].

Furthermore, our work outperforms the standard on the Italian corpus [Cresci et al., 2015] for the global classification task. We will not cover cross-event models.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Final Considerations

In this study, we presented and explored a series of consolidated and novel architectures for text classification. Starting from a (brief) introduction about social media adoption in times of necessity (section 2.3.1) and the definition of informativeness (section 3), we described the dataset of our choice (section 5.1), together with a series of preprocessing steps and offered a description of the dataset partitioning in the (sections 5.1.3, 5.2.1). In section 6 we presented the experimental settings and the results of our analysis. Finally, in section 6.3, we attempted a comparison with the topical research. The following paragraphs try to provide a meaningful starting point for future research opportunities.

Our results show that sophisticated Deep Learning architectures outperform the current state-of-the-art standards in both the supervised and semi-supervised settings. Our datasets comprise different disaster types, hence provide a heterogeneous view of disaster assessment and response on Twitter. Nonetheless, we believe that gathering disaster-specific unlabeled and/or labeled tweets would improve the performance o the classifiers. Generalizations on the nature and type of disasters could negatively affect the specificity of the disaster and its classifier, causing detrimental effects; researchers have shown that accuracy significantly drops when using pre-trained classifiers on different disasters [Imran et al., 2013b], [Fraustino et al., 2012].

7.2 Future Work

Future extensions to this work can tackle several other fronts:

- Kernelized SVMs. Although a linear kernel is often recommended for many text classification tasks (text produces a high-dimensional feature space) [Joachims, 1998], it could be interesting to try different kernels and evaluate the performance in terms of accuracy and effort (mapping to a higher dimensional space is computationally expensive).
- A set of more comprehensive features for NB, following the work [Neppalli et al., 2018].
- CNN extensions using contextualized embeddings like ELMo and Bert [Peters et al., 2018], [Devlin et al., 2018].
- More in-depth analysis of possible tuning and optimizations of the hyperparameters of the employed architectures.
- Some issues of our architectures arise with smaller portions of data. When the classifier is not strong enough, the model internalizes the label unbalance causing performance reduction. Approaches like oversampling and undersampling could help reduce the skew in the data preventing this phenomenon to occur.
- Our work also attempts to navigate the frontier of language-dependent analysis by including a form non-specificity of the referenced language. The architectural design, despite some differences (pre-trained language-specific embeddings, language-specific stopwords,

etc.) is independent of the corpus language. The next step could tackle the use of multi-lingual embedding or cross-lingual embedding models [Ruder et al., 2017], [Chen and Cardie, 2018], [Conneau et al., 2017].