Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 109C (2017) 359–366

Procedia Computer Science

www.elsevier.com/locate/procedia

The 8th International Conference on Ambient Systems, Networks and Technologies, ANT 2017

Using Word Embedding and Ensemble Learning for Highly Imbalanced Data Sentiment Analysis in Short Arabic Text

Sadam Al-Azani, El-Sayed M. El-Alfy*

Information and Computer Science Department, College of Computer Sciences and Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

Abstract

Sentiment analysis has gained increasing importance with the massive increase of online content. Although several studies have been conducted for western languages, not much has been done for the Arabic language. The purpose of this study is to compare the performance of different classifiers for polarity determination in highly imbalanced short text datasets using features learned by word embedding rather than hand-crafted features. Several base classifiers and ensembles have been investigated with and without SMOTE (Synthetic Minority Over-sampling Technique). Using a dataset of tweets in dialectical Arabic, the results show that applying word embedding with ensemble and SMOTE can achieve more than 15% improvement on average in F_1 score over the baseline, which is a weighted average of precision and recall and is considered a better performance measure than accuracy for imbalanced datasets.

Peer-review under responsibility of the Conference Program Chairs.

Keywords: Arabic sentiment analysis, Polarity classification, Word embedding, Ensemble learning, Imbalanced dataset, SMOTE, Tweets.

1. Introduction

Sentiment analysis (SA) has become a very active research area in natural language processing (NLP) and has attracted increasing interest in data mining, Web mining, and text mining [16,22]. The aim of sentiment analysis is to analyze people's opinions, evaluations, attitudes, appraisals, and emotions towards particular entities [22]. These entities might be services, organizations, products, events, topics, individuals, issues, or their attributes. Sentiment analysis includes several tasks such as opinion extraction, subjectivity classification, polarity determination, affect analysis, review mining, etc.

The research interest in sentiment analysis can be attributed to several reasons. First of all, it has a wide range of applications, and is applicable in several domains, such as branding and product analysis [20], expressive text-to-speech synthesis [4], question answering [32], analysis of political debates [9], tracking sentiment timeliness in online forums and

* Corresponding author. Tel.: +966-13-860-1930
E-mail address: alfy@kfupm.edu.sa

Peer-review under responsibility of the Conference Program Chairs.
10.1016/j.procs.2017.05.365

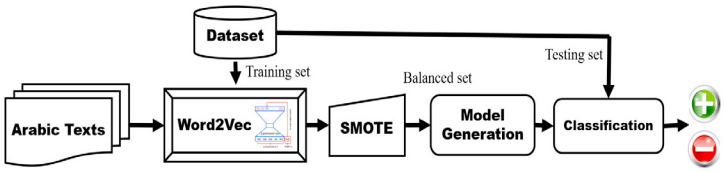

Fig. 1. Layout of the polarity determination approach

news[23], and conversation summarization[8]. Second, there are still several gaps and challenging research problems that require further studies to build more reliable and effective systems. Third, sentiment analysis is a helpful and useful tool to analyze the rapid growth of user-generated contents which are expressed in several online media such as blogs, wikis, web forums and social networks. Through these platforms or environments, users can express their opinions, post information, share knowledge, and get feedback from each others.

Approaches of sentiment analysis have been broadly categorized into: machine learning-based, lexicon-based and hybrid approaches. In the literature, it has been found that machine-learning approaches can outperform lexicon-based approaches in terms of accuracy[17]. However, their performance depends on the features extracted for a particular language and domain of application. Arabic is a Semitic language which is morphologically rich and is classified into classic Arabic, modern standard Arabic (MSA), and dialectical Arabic. Among these types, dialectical Arabic is relatively challenging due to the many varieties in existence. Most of previous work on Arabic sentiment analysis has considered either classic Arabic or modern standard Arabic based on hand-crafted features, e.g. for sentiment classification[2,3,6,13,14,26,27,29,30] or sentiment intensity[31]. However, dialectical Arabic has currently become not only used in spoken language but also in written language in different social network environments. Dealing with micro-blogs or tweets adds additional challenges since they are expressed in informal style with different spelling variations, in addition to being short text. When applying the traditional SA models, it requires tedious preprocessing operations for feature extraction. Alternatively, word embedding can alleviate this issue by using it as a source for feature engineering and extraction. Recently, it has been applied for Arabic sentiment analysis[1]. However, the experimental investigations were only conducted on balanced datasets although it is more natural to have significantly varying distributions of various opinion polarities.

The purpose of this paper is threefold. First, we investigate the performance of word embedding with various classifiers on a highly imbalanced dataset. Second, we apply SMOTE (Synthetic Minority Over-sampling Technique)[11] on the training data and compare its impact on the performance. Finally, we explore the performance of different ways of combining classifiers using ensemble learning algorithms.

The rest of this paper is organized as follows. The research methodology is described in Section 2. The experimental work and results are discussed in Section 3. Section 4 concludes the paper and highlights key findings and suggestions for future work.

2. Methodology

Figure 1 shows the layout of the proposed polarity determination approach. The following subsections present the details of each of the main operations in this block diagram.

2.1. Word Embedding

Word2vec is a powerful tool developed by Google in 2013[5,24]. It efficiently computes word vector representations in high-dimensional vector space. Word vectors are located in the vector space where words that have similar semantic and share common contexts are mapped nearby each other in the space. In addition to syntactic information, similarity of word representations obtain semantic features such that semantic relationships are often preserved in vector

operations on word vectors. For example, vector of (King) + vector of (Woman) - vector of (Man) is close to vector of (Queen). The word vector representations proved to be efficient and successful technique in the applications of NLP such as text clustering and classification and sentiment analysis. Word2vec has two neural network architectures: continuous bag-of-words (CBOW) and skip-gram. These architectures are algorithmically similar. However, CBOW architecture is trained to predict a word based on its given context whereas skip-gram predicts the context of a given word.

2.2. SMOTE

We used SMOTE to handle the problem of imbalanced dataset. SMOTE [11] over-samples the minority classes by adding synthetic samples based on feature-space similarities between existing minority examples. For each example x_i, where $x_i \in S_{min}$ (the minority class), it considers its k-nearest neighbors. SMOTE computes the difference between the sample under consideration and its nearest neighbor and multiplies it by a random number between zero and one [19]. Two most important parameters of SMOTE are the number of nearest neighbors and kind of estimators. In our experiments, we used five nearest neighbors and SVM estimator.

2.3. Model Generation and Polarity Classification

A powerful technique to improve classification performance is to combine different base classifiers trained using the same data, i.e., ensemble learning [12]. However, individual classifiers need to be combined efficiently in order to improve the performance. A necessary and sufficient condition for ensemble classifiers to perform better than base members is the accuracy and diversity of the individual classifiers [18]. The accuracy can be attained when the error rate of an individual classifier is lower than random guessing on new values [12]. On the other hand, the diversity, here, means that error rates of the base members of an ensemble are not identical [12]. For example, assume we have an input X to be classified, where $X = \{x_1, x_2, ..., x_n\}$ and n is the number of features, and we construct a voting-based ensemble classifier of three base members $\{h_1, h_2, h_3\}$. If the error rates of the base classifiers are identical, and X was classified incorrectly using the first classifier (h_1), then X will be classified incorrectly using the other two classifiers (h_2, h_3). However, if the diversity condition is attained and X was classified incorrectly using classifier (h_1), then X might be classified correctly using the other classifiers (h_2, h_3); hence, X will be classified correctly based on the majority voting [12].

Several ensemble learning techniques are evaluated to generate computational models in this paper, including: Voting, Bagging, Boosting, Stacking and Random Forests. The voting approach combines different base classifiers using hard or soft voting to predict the class labels. Hard voting considers the majority vote while soft voting decides based on the average predicted probabilities. Voting ensemble learning is recommended to balance out the individual weaknesses of a set of equally well performing models. Different combination of voting-based ensemble learning are investigated, which are discussed later. Bagging and boosting generate multiple versions of a predictor and then aggregate their individual predictions: Bagging assigns the same weight for each sample, while boosting assigns varying weights to the dataset samples. We used Gradient Boosting Classifier (GBC) and AdaBoost Classifier as instances of boosting ensemble learning. Stacking is often used to combine different classification algorithms on a single dataset. It provides the concept of a meta-learner, which is another learning algorithm; base models are trained using the training set and then the meta-learner is fitted based on the outputs of the base classifiers. Random forest is a typical example of ensemble learning for classification using a combination of decisions trees [7]. Further models are generated using single classifiers, including: k-NN, SVM, Logistic regressions, Stochastic Gradient Descent (SGD), Gaussian Naïve Bayes and Decision Trees. For SVM, we evaluated several variations including: C-Support Vector Classification (SVC), Nu-Support Vector Classification (NuSVC), and Linear Support Vector Classification (LinearSVC). SVC is implemented based on LIBSVM [10]. The fit time complexity is more than quadratic with the number of samples. This doesn't scale well for large datasets. NuSVC is similar to SVC. However, it slightly differs from SVC in that NuSVC uses a parameter to control the number of support vectors. LinearSVC differs from SVC in that it is implemented in terms of 'liblinear' [15] rather than 'libsvm'. It is more flexible than SVC in terms of the choice of penalties and loss functions. It outperforms SVC such that it scales better for large datasets.

Table 1. Distribution of positive and negative polarities in the dataset

Dataset	Positive	Negative	Total
Training	378	1060	1438
Testing	70	290	360
Total	448	1350	1798

3. Experiments and Results

We used the positive/negative polarity of the Syria_Tweets dataset[25] [1]. The dataset is highly imbalanced; the majority class is "negative", while the minority class is "positive": 1350 negative and 448 positive tweets were collected in May 2014 by polling the Twitter API for tweets originated from Syria. The dataset is expressed in Levantine dialectical Arabic and annotated manually. In our experiments, 80% of the dataset were used for training and 20% for testing. Table 1 shows the distribution negative and positive classes in the dataset.

In our study, a pretrained CBOW language model is used[1] [2]. For this language model, different experimental settings were investigated, including window sizes of 5, 10, 15, and 20 words and embedding dimensions of 200, 300, 500, and 700[1]. Eventually, a window size of 10 words with 300 dimensions was used as the best choice for learning good embeddings. The word2vec language model was learned using a large Arabic corpus of around 190 million words with compiled from various sources (Quran-text, Watan-2004, CNN-Arabic, BBC-Arabic and consumer review)[1]. The learned embedding size generated from the compiled training corpus is around 159,000 vocabulary. A feature vector is generated for each sample by averaging the embeddings of that sample.

SMOTE was applied to the training set only to ensure evaluating models using real test set not synthesized test set. Additionally, evaluating computational models using synthesized samples results in high performers which is, in fact, incorrect. SMOTE was applied using imbalanced-learn toolbox[21], which is implemented in Python[3].

The classification experiments were implemented also in Python using scikit-learn package[28]. For voting based ensemble classifier, we evaluated several combinations of base classifiers. The first combination is composed of five base classifiers, namely: SGD, RF, Logistic Regression, Logistic Regression CV and Gaussian NB. The second combination is composed of three base classifiers, SGD, RF and Logistic Regression CV. Another combination is based on SGD, RF and Linear SVC. The fourth voting-based ensemble classifier is based on SGD, Logistic Regression and Logistic Regression CV. With voting-based ensemble classifier, both hard or soft voting approaches have been considered. We evaluate Bagging Classifier using two different base-estimators: k-NN and Decision trees, the AdaBoost classifier is evaluated using Decision tree as base estimator. Table 2 summarizes the adopted parameters for each classifier.

Table 3 shows the results obtained using individual and ensemble classifiers with and without applying SMOTE. We evaluated five measures: precision (Prc), recall (Rec, accuracy (Acc), and F_1 score. Since the dataset is imbalanced, accuracy is not a good measure for performance. F_1 is a more informative score since it considers both precision and recall measures. These measures are computed as follows:

$$Prc = \frac{TP}{TP + FP} \times 100 \tag{1}$$

$$Rec = \frac{TP}{TP + FN} \times 100 \tag{2}$$

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \tag{3}$$

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

where TP, TN, FP, and FN denote true positive, true negative, false positive and false negative, respectively.

[1] The dataset is publicly available in: http://saifmohammad.com/WebPages/ArabicSA.html
[2] Available in: https://github.com/iamaziz/ar-embeddings
[3] The imbalanced-learn toolbox is publicly available in: GitHub: https://github.com/scikit-learn-contrib/imbalanced-learn

Table 2. Summary of evaluated base and ensemble classifiers

	Name	Type	Parameters
clf1	SGD	Single classifier	L2 norm regularization, Hinge loss
clf2	SGD	Single classifier	L1 norm regularization, Logistic regression loss
clf3	C-Support Vector Classification (SVC)	Single classifier	LibSVM, RBF Kernel, C = 1, gamma=0.003
clf4	SVC	Single classifier	RBF kernel, C=1, gamma=0.077
clf5	Linear SVC	Single classifier	C=10, Linear kernel
clf6	Nu-Support Vector Classification (NuSVC)	Single classifier	nu=0.5, RBF kernel
clf7	Logistic Regression (LR)	Single classifier	L2 norm regularization, Liblinear solver, C = 1
clf8	Logistic Regression CV (LRCV)	Single classifier	L2 norm regularization, Libfgs solver, Cs = 10
clf9	Gaussian NB (GNB)	Single classifier	
clf10	*k*-NN	Single classifier	$k = 1$
clf11	*k*-NN	Single classifier	$k = 3$
clf12	*k*-NN	Single classifier	$k = 5$
clf13	Decision Tree	Single classifier	Gini index, min sample split = 2
eclf1	Random Forest (RF)	Ensemble: Randomization	n_estimators=100
eclf2	Random Forest	Ensemble: Randomization	random_state=1
eclf3	Gradient Boosting	Ensemble: Boosting	
eclf4	AdaBoost	Ensemble: Boosting	Estimator: Decision Tree
eclf5	Bagging	Ensemble: Bagging	Estimator: Decision Tree
eclf6	Bagging	Ensemble: Bagging	Estimator: K-Neighbors
eclf7	voting	Ensemble: Hard voting	Base classifiers: SGD, RF, LR, LRCV and GNB
eclf8	voting	Ensemble: Soft voting	Base classifiers: SGD, RF, LR, LRCV and GNB
eclf9	voting	Ensemble: Hard voting	Base classifiers: SGD, RF and LRCV
eclf10	voting	Ensemble: Soft voting	Base classifiers: SGD, RF, and LRCV
eclf11	voting	Ensemble: Hard voting	Base classifiers: SGD, RF, and Linear SVC
eclf12	voting	Ensemble: Hard voting	Base classifiers: SGD, LR and LRCV
eclf13	voting	Ensemble: Soft voting	Base classifiers: SGD, LR and LRCV
eclf14	Stacking	Ensemble: Stacking	Base classifiers: SGD (Logistic regression loss, L1 norm regularization) and SGD(L2 norm regularization, Hinge loss), Meta-classifier: NuSVC

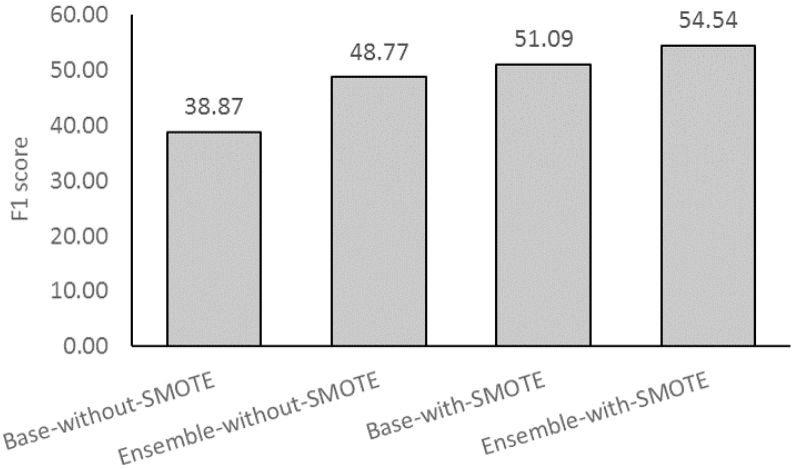

Fig. 2. Average F_1 score for various types of classifiers

The performances of the base classifiers without applying SMOTE are considered as the baselines in this paper. The highest F_1 score in the baselines (i.e., using the single classifiers without SMOTE) is 56.10% which is obtained using clf1 (SGD). The best F_1 score, however, is 63.95% which is attained for eclf14 (stacking-based ensemble) after applying SMOTE. The overall average of F_1 score before applying SMOTE is 44%, while it is 52.88% after applying SMOTE. Most single classifiers are affected positively after applying SMOTE in terms of F_1 score, except *k*-NN (when $k = 1$ or 5).

Table 3. Performance comparison of various base and ensemble classifiers with and without SMOTE in terms of precision (*Prc*), recall (*Rec*), F_1 score, and accuracy (*Acc*) (highest scores are shown in bold and lowest scores are shown with asterisk)

Classifier	Without SMOTE				With SMOTE			
	Prc	*Rec*	F_1	*Acc*	*Prc*	*Rec*	F_1	*Acc*
clf1	48.94	65.71	56.10	80.00	60.00	64.29	62.07	84.72
clf2	69.70	32.86	44.66	84.17	60.81	64.29	62.50	85.00
clf3	0.00*	0.00*	0.00*	80.56	73.08	27.14	39.58	83.89
clf4	0.00*	0.00*	0.00*	80.56	45.57	51.43	48.32	78.61
clf5	58.62	48.57	53.12	83.33	51.25	58.57	54.67	81.11
clf6	72.41	30.00	42.42	84.17	57.32	67.14	61.84	83.89
clf7	72.22	18.57	29.55	82.78	46.15	60.00	52.17	78.61
clf8	57.38	50.00	53.44	83.06	49.4	58.57	53.59	80.28
clf9	37.50	60.00	46.15	72.78	40.4	57.14	47.34	75.28
clf10	50.79	45.71	48.12	80.83	32.69	72.86	45.13	65.56
clf11	51.11	32.86	40.00	80.83	31.11	**80.00**	44.80	61.67
clf12	64.10	35.71	45.87	83.61	30.11	**80.00**	43.75	60.00*
clf13	41.38	51.43	45.86	76.39	42.86	55.71	48.45	76.94
eclf1	**75.00**	30.00	42.86	84.44	53.23	47.14	50.00	81.67
eclf2	58.54	34.29	43.24	82.50	50.00	44.29	46.97	80.56
eclf3	61.90	37.14	46.43	83.33	58.97	65.71	62.16	84.44
eclf4	38.64	48.57	43.04	75.00	39.18	54.29	45.51	74.72
eclf5	57.50	32.86	41.82	82.22	51.92	38.57	44.26	81.11
eclf6	69.57	22.86	34.41	83.06	32.95	82.86	47.15	63.89
eclf7	61.40	50.00	55.12	84.17	51.32	55.71	53.42	81.11
eclf8	62.50	50.00	55.56	84.44	47.62	57.14	51.95	79.44
eclf9	65.62	30.00	41.18	83.33	59.46	62.86	61.11	84.44
eclf10	64.81	50.00	56.45	85.00	51.02	71.43	59.52	81.11
eclf11	71.05	38.57	50.00	85.00	61.19	58.57	59.85	84.72
eclf12	60.00	51.43	55.38	83.89	51.76	62.86	56.77	81.39
eclf13	63.79	52.86	57.81	85.00	56.79	65.71	60.93	83.61
eclf14	53.41	67.14	59.49	82.22	61.04	67.14	**63.95**	**85.28**
Average	55.11	39.52	44.00	82.10	49.90	60.42	52.88	78.63

For ensemble learning methods, nearly all cases after applying SMOTE have better performance in terms of F_1 score, except the voting based ensemble classifier with five base classes (eclf7 and eclf8). It can be noticed that ensemble learning methods are more robust with imbalanced datasets than single classifiers such that the overall average of F_1 score of ensemble classifiers is 48.77%, while it is 38.87% for the single classifiers. This also holds after applying SMOTE, where the overall average of F_1 of ensemble classifiers (54.54%) is higher than for single classifiers (51.09%). It is, also, clear that stacking ensemble (eclf14) has relatively better performance than other ensemble categories (eclf1 to eclf13). Voting-based ensemble classifiers are also more robust for imbalanced dataset followed by Gradient Boosting classifier. Figure 2 shows the average F_1 scores for various types of classifiers with and without SMOTE.

4. Conclusion

This paper considers three critical issues of sentiment analysis: (a) Dealing with micro-blogging data which is characterized by its informal style and short text; (b) Handling imbalanced class distributions problem of in opinion mining; and (c) Addressing dialectical Arabic or colloquial Arabic. A well-known and mostly used learning approach for word embedding, word2vec, is applied as the main source of features. Furthermore, the problem of imbalanced dataset is addressed by over-sampling the minority class by adding synthetic samples using the SMOTE technique. The effectiveness of applying SMOTE to word embedding features is evaluated using several machine learning classifiers. Ensemble learning methods are applied to boost the performance of individual classifiers, including: Bagging, Boosting, Voting, Stacking and Random Forests. The experimental results show that applying SMOTE improves the

results in most cases and the highest overall average of F_1-score is attained when using the stacking ensemble learning method. As future work, we are working on more experiments to evaluate the performance on other datasets and other types of features.

Acknowledgment

The authors would like to thank King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia, for support during this work.

References

1. Altowayan A. Aziz and Lixin Tao. Word embeddings for arabic sentiment analysis. In *IEEE International Conference on Big Data*.
2. Mohammed N Al-Kabi, Mahmoud A Al-Ayyoub, Izzat M Alsmadi, and Heider A Wahsheh. A prototype for a standard arabic sentiment analysis corpus. *International Arab Journal of Information Technology (IAJIT)*, 13, 2016.
3. Bashar Al Shboul, Mahmoud Al-Ayyoub, and Yaser Jararweh. Multi-way sentiment classification of arabic reviews. In *Proceedings of the 6th IEEE International Conference on Information and Communication Systems (ICICS)*, pages 206–211, 2015.
4. Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. Emotions from text: machine learning for text-based emotion prediction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 579–586. Association for Computational Linguistics, 2005.
5. Tomas AMikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at International Conference on Learning Representations*, 2013.
6. Belgacem Brahimi, Mohamed Touahria, and Abdelkamel Tari. Data and text mining techniques for classifying arabic tweet polarity. *Journal of Digital Information Management*, 14(1):15, 2016.
7. Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
8. Giuseppe Carenini, Raymond T Ng, and Xiaodong Zhou. Summarizing emails with conversational cohesion and subjectivity. In *Association for Computational Linguistics*, volume 8, pages 353–361, 2008.
9. Paula Carvalho, Luís Sarmento, Jorge Teixeira, and Mário J Silva. Liars and saviors in a sentiment annotated corpus of comments to political debates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 564–568, 2011.
10. Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.
11. Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, pages 321–357, 2002.
12. Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.
13. RM Duwairi, Nizar A Ahmed, and Saleh Y Al-Rifai. Detecting sentiment embedded in arabic social media–a lexicon-based approach. *Journal of Intelligent & Fuzzy Systems*, 29(1):107–117, 2015.
14. Hady ElSahar and Samhaa R El-Beltagy. Building large arabic multi-domain resources for sentiment analysis. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 23–34, 2015.
15. Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.
16. Mohsen Farhadloo and Erik Rolland. Fundamentals of sentiment analysis and its applications. In *Sentiment Analysis and Ontology Engineering*, pages 1–24. 2016.
17. Zhang Hailong, Gan Wenyan, and Jiang Bo. Machine learning and lexicon based methods for sentiment classification: A survey. In *IEEE Web Information System and Application Conference (WISA)*, pages 262–265, 2014.
18. Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.
19. Haibo He, Edwardo Garcia, et al. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.
20. Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177, 2004.
21. Guillaume Lemaitre, Fernando Nogueira, and Christos K Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Computing Research Repository (CoRR):1609.06570*, 2016.
22. Bing Liu. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, 2012.
23. Levon Lloyd, Dimitrios Kechagias, and Steven Skiena. Lydia: A system for large-scale news analysis. In *International Symposium on String Processing and Information Retrieval*, pages 161–166, 2005.
24. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.
25. S.M. Mohammad, M. Salameh, and S. Kiritchenko. How translation alters sentiment. *Journal of Artificial Intelligence Research*, 55:95–130, 2016.

26. Mahmoud Nabil, Mohamed Aly, and Amir F Atiya. Astd: Arabic sentiment tweets dataset. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing*, pages 2515–2519, 2015.
27. Nazlia Omar, Mohammed Albared, Tareq Al-Moslmi, and Adel Al-Shabi. A comparative study of feature selection and machine learning algorithms for arabic sentiment classification. In *Asia Information Retrieval Symposium*, pages 429–443, 2014.
28. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.
29. Abdullateef M Rabab'ah, Mahmoud Al-Ayyoub, Yaser Jararweh, and Mohammed N Al-Kabi. Evaluating sentistrength for arabic sentiment analysis. In *Proceedings of the 7th IEEE International Conference on Computer Science and Information Technology (CSIT)*, pages 1–6, 2016.
30. Eshrag Refaee and Verena Rieser. An arabic twitter corpus for subjectivity and sentiment analysis. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*, pages 2268–2273, 2014.
31. Eshrag Refaee and Verena Rieser. iLab-Edinburgh at SemEval-2016 Task 7: A hybrid approach for determining sentiment intensity of Arabic twitter phrases. In *Proceedings of the 10th International Workshop on Semantic Evaluation SemEval-2016*, SemEval'16, San Diego, California, June 2016.
32. Hong Yu and Vasileios Hatzivassiloglou. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing*, pages 129–136. Association for Computational Linguistics, 2003.