Automatic Genre Classification for Resource Scarce Languages

D.P. Snyman[1], Gerhard B van Huyssteen[1] & Walter Daelemans[2]
[1]Centre for Text Technology (CTexT), North-West University, Potchefstroom, South-Africa
[2]CLiPS-CL, University of Antwerp, Antwerp, Belgium
[1]{Dirk.Snyman;Gerhard.Vanhuyssteen}@nwu.ac.za
[2]Walter.Daelemans@ua.ac.be

Abstract— **In this article we present research on the development of automatic genre classification systems for resource scarce languages. The main approaches to text classification from literature are presented and weighed against each other during an experimental phase, to identify the most appropriate text classification approach to be used as a genre classification system. A fixed feature set is extracted for seven classes from the available training data for each of the six languages under scrutiny and paired with each classification algorithm in order to test the algorithms' performance. The algorithm showing the best results is support vector machines, in conjunction with term frequency and inverse document frequency features.**

Keywords - genre classification; resource scarce languages; text classification algorithm, term frequency, inverse documents frequency.

I. INTRODUCTION

In the domain of text processing, it is often of great value to have access to metadata regarding the texts being analysed [1]. Metadata can be something as simple as an inventory containing the amount of words contained in these texts, to a summary of textual statistics. When compiling corpora for natural language processing experiments, it is vital to have this kind of information at hand as to ensure representativeness of the corpus. Experiments based on the skewed data from an unrepresentative corpus can be unreliable. However, this metadata is not available when using data from unstructured sources such as the web. Information then has to be added to the data before it can be used with some level of assurance. This can be a time consuming and expensive process. It is, however, possible to perform this automatically by implementing a text classification algorithm. Such an algorithm can append metadata as a set of predefined classes to the text, based on the analysis of its textual content, structure, style etc. [2]. These classes can be further exploited as a stepping stone for gathering extra information about the data, or can be used in its original state as metadata.

An example of an application of the abovementioned text classification algorithms is to classify a text, based on its genre [3]. Genre classification is not to be confused with topic-based classification (although there are some similarities in classification methods). Consider a feature article: While the topic of such an article could be about any given subject, the genre of the text remains that of a feature article. The same holds true for any other genre as they are essentially topic independent even though some topics are more prevalent in some genres than others. This manner of classification is used in a number of commonplace systems. These include automatic document management systems [4], email classifiers/spam filters [4] and Automatic sentiment analysis [4].

It would be expected that these and similar systems are made available in regionally appropriate modes (i.e. compatible languages) but this unfortunately is not the case, especially for resource scarce or underrepresented languages. To our knowledge there are no genre classification systems available for the South African languages. This poses a significant dilemma for the development of text resources for these languages, particularly where representativeness across genres is of great concern. In a project for the National Centre for Human Language Technologies it was seen that it would indeed be necessary to classify texts according to their genre before these texts can be included in representative corpora and because of the sheer volume of texts that have to be analysed, an automatic solution is required In a project undertaken for the department of Arts and Culture of the South African government, this need is addressed in order to be able to provide quality resources.

We therefore present research on determining the best text classification algorithm, to be used in a genre classification system, for six of the South African languages. This core technology/classifier can then be used as the basis for constructing some of the abovementioned applications, or as a standalone system for genre classification of text data prior to corpus construction.

II. RELATED WORK

Relatively little research has been done on the topic of genre classification in the last two decades [5][8], because collections of texts have, until recently, usually been homogeneous in their compilation [5] i.e. domain specific. This compilation has changed to a more diverse one as the need arose for more representation across domains and text types. As a result, research has recently gained momentum, encouraged by the alarming increase in the amounts of digitalised documents and data that somehow need to be managed [6]. Data management systems have, for the most part, been concerned with classification based on the topic of texts. However, users rarely require information based on topics alone [5][8]. They require different styles of text for different practices e.g. editorials, scholarly publications,

novels, etc. Therefore by adding a genre classification to texts, these requirements can be met.

A. Resource scarcity

Our research focuses on languages that have little annotated resources available, even for casual experimenting. Resource scarceness with regards to this kind of application, to our knowledge, isn't discussed in literature. We therefore have no reference as to the level of performance to expect from such a classification algorithm based on these kinds of limited resources. We therefore propose to compare these classifiers to state of the art systems for languages with abundant data available. This will also indicate the level of success associated with the selection of the algorithm for genre classification if the resource scarce counterpart can (to some extent) match the performance of the established classifiers.

B. State of the art

Yi-Hsing and Hsiu-Yi [7] present the development of a genre classification system for the classification of 11 web genres. Along with the classification algorithm, they make use of a domain ontology table to establish relationships between text tokens which are then relayed to the classification algorithm (Bayesian classifier) as weights associated with each relationship. This genre classification system is then implemented as part of a larger document management system. They report accuracies ranging from 60% to 89% across all of the 11 classes.

In an attempt to determine the best possible combination of feature sets, extracted from 1224 web documents, Lim et al. [11] reports an average accuracy across 16 classes of 75.7% on their best possible combination of features. They use html-based tags, url info, lexical features and structural information to compile the feature sets. For the classification algorithm they make use of Timbl's *k*-nn implementation, where *k*=1.

Fin and Kunshmerick [12] weigh the impact of three sets of features for genre classification of texts, having to distinguish between opinionated and factual articles across three domains. Using a corpus of approximately 800 texts, and extracting features like part-of-speech information, a bag of words approach, textual statistics and a combination of all the aforementioned features, they record average accuracy scores between 82.4% and 90.5%.

These state of the art examples from literature, should give a good indication of the kind of results that are to be expected after evaluation, taking into account the lack of resources available.

III. Experimental setup

WEKA [9] is a suite of machine learning algorithms offered as an experimental environment. Due to the nature of the research we opted for using this kind of setup where these algorithms can be easily executed and compared to each other. It also holds the benefit of providing access to pre-processing scripts for text to vector conversion with a range of feature extraction options.

A. Algorith selection

Throughout literature, there are a reoccurring set of algorithms that are generally used for text classification experiments: *K-nearest neighbour, decision trees, Naive Bayes, multinomial Naive Bayes* and *support vector machines* [2][3][5][6][7]. We will consequently train each of these classifiers and compare their results.

B. Data

The training data for each language is derived from parallel documents for which the genre classes are already defined in Afrikaans. By knowing/defining the class of each document in a known language, the parallel versions in the other languages, can be automatically classified as having the same class. These classes are defined in Table 1. These texts have been extracted from public domain government websites and cleaned from all irregular characters. The texts are converted to plain text format and the language of each text is determined automatically by using an automatic language identification system as described in [14].

Because of the data scarceness of the languages and some being scarcer than others, we see a big difference in the number of available textual units for each language (Table 1). This has also resulted in some of the South African languages being left out altogether, as there were no available documents to be used as training data. The languages that are present are abbreviated according to the ISO 639-2 standard, as follows: af – Afrikaans, nso – Sepedi, st – Sesotho, tn – Setswana, xh – isiXhosa and zu – isiZulu. McCallum and Nigam [10] state the use of training sets containing any number of texts between 4000 and 15000. Using these numbers as a guideline for the number of training instances required for text classification, we see that it is significantly more than what we have available for the resource scarce languages. We therefore do not expect similar results upon evaluation of our systems, but should be able to judge their performance while keeping the data scarceness in mind.

			Number of texts per language					
Class	**Label**	**Description**	**af**	**nso**	**st**	**tn**	**xh**	**zu**
Advertisement	ADV	Advertisements in papers/internet.	346	9	21	7	133	14
Information	INF	Informational pamphlets.	394	385	392	388	352	15
Instruction	INS	Manuals, directions.	564	546	548	543	100	390
News	NEW	Reporting.	412	45	48	211	384	540
Official texts	OFF	Bills, laws, policies.	108	8	19	11	48	5
Poetry	POE	Poems, lyrics.	528	28	37	29	22	27
Speech	SPE	Debates, public addresses.	144	55	87	50	49	57

Table 1. Classes for classification algorithm and number of instances per class and language

Before training the classifiers, the data is passed through WEKA's pre-processing scripts which construct document vectors from extracted features. We chose to limit the feature sets to term frequency and inverse document frequency scores [10]. Term frequency (*tf*) is the number of occurrences of a token (word) noted across the entire collection of documents in the training corpus. Inverse document frequency (*idf*) is the number of documents in the training corpus containing a specific token, inversed. The product of these two scores, gives an indication of the contribution made by a specific token to the class of the document containing the token. Equation (1) is used to determine *tf-idf* scores.

$$\mathbf{(tf\text{-}idf)}_{i,j} = \mathbf{tf}_{i,j} \times \mathbf{idf}_{i,j} \quad (1)$$

The choice to limit features to *tf-idf* is due to the unavailability of other textual information, which cannot be readily extracted without some pre-existing system and/or knowledge about the languages in question, and due to the focus of the research being algorithm selection rather than feature optimisation. All the languages are evaluated separately across the collection of algorithms and we expect the features to mitigate any language specific issues which may affect the performance of the algorithms, because of .

According to McCallum and Nigam [10], their larger sets are tokenized and focussed by stemming and removing low frequency words as well as words contained in stop lists, while smaller sets are used in their entirety. Stemming and word deletions can actually hamper performance on smaller training sets as the critical mass needed for accurate classification is lost [10].The training data is therefore only normalised by converting all the tokens in the training set to lowercase to prevent the abovementioned losses.

C. Evaluation

The standard information retrieval measures, precision, recall and F1-measure are used to evaluate the effectiveness of classification for the system [2][5][6][8][7][9][10]. This is represented in Table 2. The evaluation method used is *n*-fold cross validation, with 90% of the data used for training, and 10% used for testing. This split is done on a per class basis, ensuring representativeness of each class during evaluation. Where a class has less than 10 textual units available in total, it would result in fractions of documents being required if 10% is to be used as training data. In these cases the evaluation is done by using one complete unit of text as the test set, and rotating the document used for each fold, minimizing the reoccurrence of one given test set. When using only fractions of these textual units, it is possible that important information that contributes to the correct classification of a text's genre is lost when the test document is pruned to the desired size.

Class C_i		**Actual Class**	
		Yes	No
Classifier class	Yes	TP	FP
	No	FN	TN

Table 2. Standard information retrieval methods [7]

The formulas for Recall, Precision, and F1-Measure of Ci (see Table 2) are shown in the following three equations (2)(3)(4), Where TP = True Positive, TN = True Negative, FN = False Negative and FP = False Positive classifications.

$$\mathbf{R\ (Recall) = TPi / TPi + FNi}, \quad (2)$$

$$\mathbf{P\ (Precision) = TPi / TPi + FPi}, \quad (3)$$

$$\mathbf{F1\ (F1\text{-}Measure) = 2(R*P) / (R+P)} \quad (4)$$

The abovementioned algorithms will be trained with the same training data and feature sets for each language, and then weighed against each other to determine the most suitable algorithm for genre classification. By keeping the training data and features constant throughout the evaluation, we ensure that any differences in performance are purely down to the differences between the algorithms and not due to the differences in features. All the algorithms are evaluated on their default settings as available in WEKA. No parameter optimisation is done to improve the results obtained as we attempt to identify the algorithm which initially indicates a good level of suitability for genre classification. This is due to the early stages of this study where exhaustive searches of parameter spaces would be too time-consuming. We will attempt to optimise these parameters and extract other textual features which could aid in algorithm performance at a later stage as part of our future work.

IV. Results

The results for the evaluation phase of our research are presented in Table 3. This shows the results for each class, language and classification algorithm combination. Weighted averages are provided by WEKA after evaluation and are also provided. Weights are assigned to each class depending on the amount of available training instances that were provided. The weights are assigned by getting the sum of predicted scores (precision, recall and F-measure are all evaluated seperately) for every element of each class and the coefficient of the number of instances is then used to calculate the weighted scores per class. These resulting scores are then used to determine the final averaged weighted scores for each algorithm, relative to the quantity of training data available. This metric is useful for our research as it provides intrinsic evaluation possibilities. This provides us with insight as to a classification algorithm's ability to cope with a lack of training data. Consider Table 3. When comparing the average weighted F1-Measures for each classification algorithm, across all of the languages, we see that in four out of the six languages, support vector machines show the best results, with naive Bayes and multinomial naive Bayes, each outperforming support vector machines only by a small margin for two of the languages.

V. Conclusion and Future Work

In this article we investigated a text classification algorithm to form the basis of a genre classifier for resource scarce languages by comparing five generally used algorithms with a fixed set of training data through experimental evaluations, in an attempt to identify the best algorithm for the classification assignment.

We see an encouraging set of scores for all of the algorithms tested. Even in the some of the unweighted results per class, we see scores similar to the state of the art systems described in Section II. We see surprisingly high scores (some attaining scores of up to 100% for precision or recall or both) for classes which have the smallest sets of training data available. We attribute this to overfitting, as Liu et al. [13] state that small classes can lead to overfitting where they are confronted with a large search space like multi-class cross validation. They emphasise this with results from machine learning experiments on automobile features, where the overfitted classes attempt to use vehicle colour to try and determine the expected gas mileage in cross validation evaluations. This attempt is logically flawed as vehicle colour could in fact have no influence what so ever on a vehicle's fuel consumption [13]. It would be expected that the same, illogical type of feature use is attempted when determining the genre of a text, due to the lack of other, more informative features available for the underrepresented classes. The main occurrence noted is, on the other hand, erratic differences in the recorded scores per class. These anomalies are attributed to the lack of proper data due to the resource scarceness of the languages under evaluation. This would make the implementation of a genre classification algorithm, supplied with our limited data, unadvisable.

The results for our investigation would (at least for the moment) suggest that support vector machines would be the sure algorithm choice for genre classification for research scarce languages (having outperformed the other algorithms in most cases). This is also the case with the evaluations for the Afrikaans training sets that have a better representation across classes, which would suggest this trend would continue when increasing the size or features of the training sets. This is in line with the results seen in [13], where support vector machines show increasingly better results than naive Bayes (and possibly other algorithms), when supplied with more and better features and training data.

We would like to verify the above findings in our future work, by further developing training sets for these resource scarce languages. By increasing the training set sizes we will attempt to mitigate the overfitting seen in underrepresented classes, as well as the erratic results seen between classes, and submit the theory that support vector machines will continue to outperform the other algorithms, to further scrutiny. Plotting learning curves for the algorithms could also provide a better insight into the training set sizes required for effective classification. We also intend to apply "intelligent" feature extraction methods in order to gauge the effect of supplying better features to the algorithms. By using algorithm optimisation to determine the best possible combination of parameters for each algorithm, we would be able to identify the best algorithm for genre classification with a greater level of certainty than we are currently able to do.

REFERENCES

[1] Cardinaels, K., Meire, M., Duval, E. 2005. "Automating metadata generation: the simple indexing interface". (In Proceedings of WWW 2005: 14th international conference on World Wide Web).

[2] Goller, C., Löning, J., Will, T., Wolff, W. 2000. "Automatic document classification: A thorough evaluation of various methods". (In proceedings of the Internationales Symposium für Informationswissenschaft (IS12000)). p. 1.

[3] Raaf, S. 2008. Automatic Genre Classification of English Students' Argumentative Essays using Support Vector Machines. North West University – Potchefstroom Campus. Potchefstroom. pp. ii-iii.

[4] Manning, C.D., Prabhakar, R., Schütze, H. 2009. An Introduction to Information Retrieval. Cambridge University Press. Cambridge, England. pp. 117-119, 253-285.

[5] Kessler, B., Nunberg, G., and Schuetze, H. 1997. "Automatic detection of text genre". (In Proceedings of ACL-97, 35th Annual Meeting of the Association for Computational Linguistics. Madrid, ES.), pp. 32–38.

[6] Sebastiani, F. 2001. "Machine learning in automated text categorisation". Technical Report IEI-B4-31-1999, Istituto di Elaborazione dell'Informazione.

[7] Yi-Hsing, C., Hsiu-Yi, H. 2008. "An Automatic Document Classifier System based on Naïve Bayes Classifier and Ontology". (In proceedings of the Seventh International Conference on Machine Learning and Cybernetics. Kunming).

[8] Liu, R., Jiang, M., Tie, Z. 2009. "Automatic Genre Classification by Using Co-training". (In proceedings of the Sixth International Conference on Fuzzy Systems and Knowledge Discovery.) Vol. 1, pp.129-132

[9] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H. 2009. The WEKA Data Mining Software: An Update. SIGKDD Explorations. Volume 11, Issue 1.

[10] McCallum, A., Nigam, K. 1998. "A comparison of event models for Naive Bayes text classification". (In Proceedings of the AAAI-98 Workshop on Learning for Text Categorization.) pp. 41-48. http://www.cs.cmu.edu/~knigam/papers/multinomial-aaaiws98.pdf. [Date of access: 2010-09-01].

[11] Lim, C. S., Lee, K. J. Lee, Kim, G. C. Kim. 2005. Multiple sets of features for automatic genre classification of web documents. Information processing and management 41(5). pp. 1263-1276.

[12] Finn, A., Kushmerick, N. 2006. "Learning to classify documents according to genre". Journal of the American Society for Information Science and Technology (JASIST), vol. 7. (Special issue on computational analysis of style).

[13] Liu, H.; Dougherty, E.R.; Dy, J.G.; Torkkola, K.; Tuv, E.; Peng, H.; Ding, C.; Long, F.; Berens, M.; Parsons, L.; Zhao, Z.; Yu, L.; Forman, G.; , "Evolving feature selection". *Intelligent Systems, IEEE* , vol.20, no.6, pp. 64- 76, Nov.-Dec. 2005. doi:10.1109/MIS.2005.105 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1556517&isnumber=33103

[14] Pienaar, W., Snyman, D.P. 2010. "Spelling Checker-based Language Identification for the Eleven Official South African Languages". (In Proceedings of the Twenty-First Annual Symposium of the Pattern Recognition Association of South Africa). 22-23 November 2010. Stellenbosch, South Africa. pp. 213 -217.

		af			nso			st		
		Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Naive Bayes	ADV	0.825	0.962	0.888	0.625	0.556	0.588	0.195	0.714	0.306
	INF	0.757	0.234	0.357	0.699	0.566	0.626	0.7	0.554	0.618
	INS	0.22	0.304	0.256	0.825	0.832	0.829	0.826	0.846	0.836
	NEW	0.41	0.671	0.509	0.388	0.889	0.541	0.279	0.813	0.415
	OFF	0.613	0.596	0.605	0.333	0.25	0.286	0.143	0.368	0.206
	POE	0.99	0.761	0.861	0.581	0.643	0.61	0.518	0.461	0.488
	SPE	0.721	0.892	0.798	0.5	0.614	0.551	0.798	0.735	0.765
	Weighted Avg.	**0.777**	**0.713**	**0.712**	**0.735**	**0.718**	**0.719**	**0.719**	**0.684**	**0.695**
Multinomial NB	ADV	0.994	0.959	0.976	1	1	1	0.552	0.762	0.64
	INF	0.675	0.841	0.749	0.797	0.418	0.549	0.751	0.492	0.595
	INS	0.292	0.076	0.121	0.755	0.917	0.828	0.784	0.91	0.842
	NEW	0.773	0.4	0.527	0.4	0.933	0.56	0.289	0.813	0.426
	OFF	0.727	0.853	0.785	0.625	0.625	0.625	0.275	0.579	0.373
	POE	0.964	0.953	0.958	0.607	0.607	0.607	0.617	0.523	0.566
	SPE	0.682	0.926	0.785	0.6	0.682	0.638	0.821	0.81	0.816
	Weighted Avg.	**0.84**	**0.841**	**0.821**	**0.746**	**0.718**	**0.702**	**0.748**	**0.729**	**0.729**
Decision Trees	ADV	0.897	0.942	0.919	0.429	0.333	0.375	0.435	0.476	0.455
	INF	0.634	0.638	0.636	0.654	0.649	0.652	0.662	0.684	0.673
	INS	0.232	0.207	0.218	0.811	0.838	0.824	0.821	0.817	0.819
	NEW	0.489	0.506	0.497	0.386	0.378	0.382	0.388	0.396	0.392
	OFF	0.648	0.541	0.59	0.6	0.375	0.462	0.429	0.158	0.231
	POE	0.87	0.9	0.885	0.444	0.429	0.436	0.54	0.539	0.54
	SPE	0.701	0.649	0.674	0.639	0.523	0.575	0.776	0.776	0.776
	Weighted Avg.	**0.751**	**0.756**	**0.753**	**0.715**	**0.718**	**0.716**	**0.714**	**0.715**	**0.714**
K-NN	ADV	0.227	0.994	0.37	1	0.444	0.615	0.727	0.381	0.5
	INF	0.73	0.356	0.479	0.604	0.706	0.651	0.611	0.633	0.622
	INS	0.333	0.152	0.209	0.804	0.77	0.786	0.814	0.785	0.799
	NEW	0.688	0.388	0.496	0.613	0.422	0.5	0.359	0.292	0.322
	OFF	0.821	0.422	0.558	0.4	0.25	0.308	0.176	0.316	0.226
	POE	1	0.021	0.041	0.375	0.321	0.346	0.582	0.512	0.545
	SPE	0.901	0.493	0.638	0.607	0.386	0.472	0.765	0.812	0.788
	Weighted Avg.	**0.751**	**0.4**	**0.377**	**0.703**	**0.698**	**0.696**	**0.706**	**0.706**	**0.705**
SVM	ADV	0.92	0.977	0.948	1	0.556	0.714	0.818	0.429	0.563
	INF	0.663	0.695	0.678	0.671	0.745	0.706	0.639	0.691	0.664
	INS	0.264	0.207	0.232	0.824	0.82	0.822	0.828	0.811	0.819
	NEW	0.582	0.541	0.561	0.784	0.644	0.707	0.61	0.521	0.562
	OFF	0.8	0.771	0.785	0.667	0.25	0.364	0.417	0.263	0.323
	POE	0.968	0.983	0.976	0.632	0.429	0.511	0.637	0.625	0.631
	SPE	0.856	0.845	0.85	0.767	0.523	0.622	0.831	0.841	0.836
	Weighted Avg.	**0.841**	**0.848**	**0.844**	**0.76**	**0.756**	**0.754**	**0.757**	**0.757**	**0.756**

Table 3. Results for evaluation on language and algorithm pairs

		tn			xh			zu		
		Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Naive Bayes	ADV	0.714	0.714	0.714	0.186	0.714	0.295	1	0.4	0.571
	INF	0.726	0.559	0.632	0.613	0.375	0.465	0.688	0.554	0.614
	INS	0.83	0.831	0.831	0.821	0.432	0.566	0.82	0.847	0.833
	NEW	0.836	0.891	0.862	0.251	0.329	0.285	0.299	0.872	0.446
	OFF	0.4	0.182	0.25	0.086	0.354	0.138	0.171	0.5	0.255
	POE	0.241	0.655	0.352	0.433	0.564	0.49	0.613	0.474	0.535
	SPE	0.362	0.641	0.463	0.715	0.727	0.721	0.787	0.799	0.793
	Weighted Avg.	**0.765**	**0.739**	**0.744**	**0.611**	**0.476**	**0.507**	**0.733**	**0.71**	**0.714**
Multinomial NB	ADV	0.7	1	0.824	0.208	0.722	0.323	0.625	1	0.769
	INF	0.724	0.534	0.614	0.646	0.42	0.509	0.751	0.426	0.543
	INS	0.768	0.896	0.827	0.776	0.545	0.64	0.775	0.894	0.83
	NEW	0.91	0.867	0.888	0.311	0.488	0.38	0.279	0.872	0.423
	OFF	0.429	0.273	0.333	0.182	0.167	0.174	0.156	0.5	0.237
	POE	0.5	0.655	0.567	0.546	0.556	0.551	0.659	0.445	0.532
	SPE	0.472	0.641	0.543	0.77	0.727	0.748	0.784	0.838	0.81
	Weighted Avg.	**0.759**	**0.758**	**0.751**	**0.631**	**0.543**	**0.566**	**0.737**	**0.707**	**0.703**
Decision Trees	ADV	0.75	0.857	0.8	0.349	0.338	0.344	0.571	0.4	0.471
	INF	0.657	0.67	0.663	0.517	0.518	0.517	0.638	0.682	0.659
	INS	0.817	0.82	0.819	0.671	0.68	0.675	0.81	0.778	0.794
	NEW	0.808	0.839	0.823	0.344	0.358	0.351	0.412	0.447	0.429
	OFF	0.4	0.182	0.25	0.297	0.229	0.259	0.273	0.214	0.24
	POE	0.444	0.276	0.34	0.522	0.529	0.525	0.518	0.515	0.516
	SPE	0.432	0.41	0.421	0.768	0.741	0.754	0.743	0.741	0.742
	Weighted Avg.	**0.74**	**0.744**	**0.741**	**0.578**	**0.577**	**0.577**	**0.694**	**0.692**	**0.693**
K-NN	ADV	1	0.571	0.727	0.372	0.338	0.354	1	0.2	0.333
	INF	0.622	0.644	0.633	0.472	0.569	0.516	0.554	0.615	0.583
	INS	0.793	0.787	0.79	0.674	0.649	0.661	0.797	0.717	0.755
	NEW	0.775	0.834	0.804	0.424	0.371	0.396	0.643	0.191	0.295
	OFF	0.4	0.182	0.25	0.36	0.188	0.247	0.286	0.143	0.19
	POE	0.273	0.207	0.235	0.484	0.556	0.518	0.612	0.449	0.518
	SPE	0.517	0.385	0.441	0.73	0.66	0.693	0.673	0.831	0.744
	Weighted Avg.	**0.712**	**0.717**	**0.713**	**0.572**	**0.568**	**0.567**	**0.673**	**0.67**	**0.661**
SVM	ADV	1	0.714	0.833	0.4	0.391	0.395	1	0.3	0.462
	INF	0.637	0.711	0.672	0.505	0.593	0.545	0.654	0.726	0.688
	INS	0.823	0.8	0.811	0.693	0.662	0.677	0.835	0.801	0.817
	NEW	0.855	0.891	0.872	0.431	0.399	0.415	0.611	0.468	0.53
	OFF	0.667	0.182	0.286	0.4	0.208	0.274	0.6	0.214	0.316
	POE	0.417	0.172	0.244	0.595	0.587	0.591	0.6	0.214	0.316
	SPE	0.704	0.487	0.576	0.8	0.785	0.792	0.79	0.822	0.806
	Weighted Avg.	**0.756**	**0.757**	**0.752**	**0.609**	**0.607**	**0.606**	**0.751**	**0.749**	**0.747**

Table 3 (Continued). Results for evaluation on language and algorithm pairs