A Study of SVM Kernel Functions for Sensitivity Classification Ensembles with POS Sequences

Graham McDonald
University of Glasgow
Glasgow, Scotland, UK
G.McDonald.1@research.gla.ac.uk

Nicolás García-Pedrajas
University of Córdoba
Córdoba, Spain
npedrajas@uco.es

Craig Macdonald
University of Glasgow
Glasgow, Scotland, UK
Craig.Macdonald@glasgow.ac.uk

Iadh Ounis
University of Glasgow
Glasgow, Scotland, UK
Iadh.Ounis@glasgow.ac.uk

ABSTRACT

Freedom of Information (FOI) laws legislate that government documents should be *opened* to the public. However, many government documents contain *sensitive* information, such as *confidential* information, that is exempt from release. Therefore, government documents must be *sensitivity reviewed* prior to release, to identify and *close* any sensitive information. With the adoption of born-digital documents, such as email, there is a need for automatic sensitivity classification to assist *digital sensitivity review*. SVM classifiers and Part-of-Speech sequences have separately been shown to be promising for sensitivity classification. However, sequence classification methodologies, and specifically SVM kernel functions, have not been fully investigated for sensitivity classification. Therefore, in this work, we present an evaluation of five SVM kernel functions for sensitivity classification using POS sequences. Moreover, we show that an *ensemble* classifier that combines POS sequence classification with text classification can significantly improve sensitivity classification effectiveness (+6.09% F_2) compared with a text classification baseline, according to McNemar's test of significance.

1 INTRODUCTION

Freedom of Information (FOI) laws state that government documents should be open to the public. However, many government documents contain *sensitive* information, such as *confidential* information. Therefore, FOI laws exempt sensitive information from release and government documents must be *sensitivity reviewed* prior to release, to identify and *close* any sensitivities. However, with the introduction of *born-digital* documents, such as email, the volume of documents has increased and document creation processes have become less structured. Hence, traditional sensitivity review processes are not viable for *digital sensitivity review*.

Automatic classification techniques can potentially be adapted to assist the digital sensitivity review process and reduce the time taken to review documents. McDonald *et al.* [12] showed that sensitivities relating to *information supplied in confidence* could be captured in the grammatical structure of documents, by representing the documents as sequences of Part-of-Speech (POS) n-grams [11]. For example, sensitivities relating to information supplied in confidence are often recounts of dialogues or actions and, therefore, can contain strings such as "an informer gave him", "the ambassador said she" or "a detainee showed us". These strings can all be represented by the POS sequence **DT NN VB PR**, or subsequently as a sequence of POS n-grams, e.g. as POS 2-grams **DTNN NNVB VBPR**. McDonald *et al.* [12] showed that the frequencies of certain POS sequences can be an indicator of potential sensitivity.

Representing documents by an abstraction, such as the POS tags they contain, has an additional attractive by-product. In effect, a document's tokens (POS n-grams) can be viewed as a sequence of symbols from an alphabet, rather than terms from a vocabulary and, hence, gives rise to the possibility of developing techniques based on sequence classification [18]. Sequence classification has been shown to be effective in fields such as Bioinformatics (e.g. classifying protein sequences) and Cyber-Security (e.g. intrusion detection), in addition to Information Retrieval (IR) tasks (e.g. bot detection from query log sequences). An intrinsic component of sequence classification is selecting a classification *kernel function* that is suitable for the classification task being attempted, for example, sequence-similarity kernels such as the Spectrum kernel [10].

Our contributions in this work are two-fold. Firstly, we present a thorough evaluation of five SVM kernel functions for POS sequence classification of sensitive information that would be exempt from release under UK FOI laws. Secondly, we show that a weighted majority vote *ensemble* classifier that combines POS sequence classification with text classification can significantly improve sensitivity classification (+6.09% F_2) compared to a text classification baseline, according to McNemar's test of significance.

This paper is structured as follows: Section 2 discusses prior work; Section 3 provides an overview of the kernel functions that we deploy; We present our experimental setup in Section 4 and our results in Section 5; Concluding remarks follow in Section 6.

2 RELATED WORK

Most of the existing literature on automatically identifying sensitive information has addressed the task of *masking* personal data [2, 5]. However, sensitive information in government documents is more

SIGIR '17, August 07-11, 2017, Shinjuku, Tokyo, Japan

978-1-4503-5022-8/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3077136.3080731

wide-ranging than personal information and can include, for example, issues of confidentiality or international relations. Gollins *et al.* [6] posited that IR technologies could assist the digital sensitivity review process. They also noted that some sensitivities, such as international relations, can pose more of a risk due to the potential effect of accidental release. Hence, there is a need for automatic techniques for classifying these more wide-ranging types of sensitivity.

Text classification has been shown to be an effective approach as a basis for automatic sensitivity classifiers [1, 13]. Text classification relies on there being a specific set of terms, for which their distribution can be a reliable indicator of the class that is to be identified. However, as Gollins *et al.* [6] noted, sensitivity arises not only from the terms in a document but also from the *context* in which they appear and, therefore, sensitivity classification must go beyond term features (and text classification). In this work, we focus on combining text classification with sequence classification techniques for sensitivity identification.

McDonald *et al.* [12] showed that Part-of-Speech (POS) *n*-gram sequences can be effective for identifying *supplied in confidence* sensitivities. They adapted an approach from Lioma and Ounis [11], who showed that more frequent POS *n*-grams in a collection are likely to bear more *content.* McDonald *et al.* used the distributions of POS *n*-grams in sensitive and non-sensitive text to measure the *sensitivity load* of text sequences. Differently from the work of [12], in this work, we use POS sequences to study different SVM kernel functions for sensitivity classification. Moreover, we investigate methods of ensemble learning for effectively combining POS sequence classification with text classification for sensitivity.

Ensemble classification [3] methods combine the decisions from a *committee* of individual classifiers with a view to improving the overall classification performance. The simplest, but often most effective, of these approaches combines the predictions from the committee classifiers by viewing each classifier's prediction as a vote for the class of a document [9]. Another popular approach, namely stacking [17], is to learn a separate (*meta-learner*) combiner function from the predictions of the committee classifiers. In this work, we investigate weighted voting and stacking ensembles for combining sequence and text classification for sensitivity classification.

3 SVM KERNEL FUNCTIONS

As previously stated in Section 1, an intrinsic component of a new sequence classification task is to identify a suitable kernel function for the task. Therefore, in this section we provide an overview of the kernel functions and classifier that we deploy for POS sequence classification for sensitivity.

Support Vector Machines (SVM) [16] are a type of supervised learning algorithm that learn a linear separating hyperplane between two classes within a vector space. SVM achieves this by solving a dual optimisation problem on a set S of training *instance* vectors, x_i, with corresponding class labels, y_i, where $i = 1..m$, $x_i \in \mathbb{R}^n$ and $y_i \in \{\pm 1\}$. The SVM optimisation aims to 1) maximise the *distance* between the hyperplane and the *closest* instances in either of the classes, and, 2) minimise the classification error. The resulting optimisation problem, $Maximise \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \left\langle x_i, x_j \right\rangle$, requires learning the optimal weights, α_i for $i = 1..m$, where $\alpha_i \geq 0$. Since this optimisation problem relies only on the inner products $\left\langle x_i, x_j \right\rangle$, which can be viewed as a distance measure, this component of the optimisation can be *substituted* by a *kernel function*, $K(x_i, x_j)$, that computes a measure that is selected for the classification task.

The linear kernel, defined as $K_{linear}(x_i, x_j) = x_i^T x_j$, is the simplest kernel. However, K_{linear} has desirable properties in that it is very fast to train and does not tend to *over-fit* the learned model to S when $|x|$ is very large [7]. For non-linearly separable data, a more suitable kernel is the Gaussian kernel, $K_{gaussian}(x_i, x_j) = exp\left(\frac{-||x_i - x_j||^2}{2\sigma^2}\right)$, where σ is a parameter that determines the *width* of the Gaussian function, i.e. the region of influence for an instance in vector space. A properly tuned Gaussian kernel will always be able to learn the optimal decision of a linear kernel [8], yet tuning σ can be expensive and does not guarantee obtaining a better model.

By substituting $\left\langle x_i, x_j \right\rangle$ with a kernel function, we effectively create a *feature* map, ϕ, which maps an instance, x, to a new (possibly higher dimensional) space. For the linear and Gaussian kernels, ϕ is implicit within the dot products defined in the functions. Often, however, kernels explicitly define this mapping as the input to the kernel function. String kernels operate on finite sub-sequences of strings and the Spectrum kernel [10] is a simple string kernel defined by its map ϕ over all sub-sequences in an alphabet A. For a given alphabet A, $|A| = l$, a document's feature map, $\Phi_k(x) = (\phi_a(x))_{a \in A^k}$, is the frequency weighted set of all contiguous subsequences of length $k \geq 1$, that the document contains, i.e. its k-spectrum, and where $\phi_a(x)$ is the frequency of a in x. The Spectrum kernel is then defined as $K_{spectrum}(x, y) = \langle \Phi_k(x), \Phi_k(y) \rangle$.

One limitation of the Spectrum kernel is that it is constrained to exact matches when calculating the similarity of instances. The Mismatch Kernel [4] addresses this by allowing for a pre-defined number of mismatched symbols within sequences. For a given sequence $\alpha = a_1..a_k, a \in A$, $N(k, m)(\alpha)$ is the set of all k-length sequences, $\beta = b_1..b_k, b \in A$ that differ from α by $\leq m$ mismatches. The Mismatch kernel's feature map is then defined as $\Phi_{(k,m)}(\alpha) = (\phi_\beta(\alpha))_{\beta \in A^k}$, where $\phi_\beta(\alpha) = 1$ if $\beta \in N(k, m)(\alpha)$, else $\phi_\beta(\alpha) = 0$. From this feature map, the (k,m)-mismatch kernel is defined as $K_{(k,m)}(x, y) = \left\langle \Phi_{(k,m)}(x), \Phi_{(k,m)}(y) \right\rangle$.

Finally, the Smith-Waterman kernel, K_{sw}, is based on the Smith-Waterman sequence similarity algorithm [15]. Unlike the kernels presented thus far, it is not strictly a kernel function, since it does not satisfy certain mathematical conditions, e.g. it is not always positive definite. However, in this work, we test its effectiveness as a kernel function for POS n-gram sequence classification.

For complex sequence classification tasks, a single SVM kernel may not provide an optimal solution. One method of addressing this is to combine multiple simpler kernels as a hybrid kernel, with the aim of considering multiple aspects of an instance vector. We hypothesise that different types of kernels will identify different aspects of sensitivity and, therefore, in this work, we evaluate two hybrid kernels that are a linear combination of the scores from two simpler kernels, namely Spectrum+Linear and Spectrum+Gaussian.

4 EXPERIMENTAL SETUP

Collection: Our test collection is 3801 government documents that have been sensitivity reviewed by government reviewers. All documents that contain any sensitivity relating to *Personal Information* or *International Relations* FOI exemptions were labeled as *sensitive*. All other documents were labeled *not-sensitive*, resulting in a binary classification task with 502 *sensitive* and 3299 *not-sensitive*

Table 1: The total unique POS n-gram tokens in each collection representation.

	1-gram	2-gram	3-gram	4-gram	5-gram	6-gram	7-gram	8-gram	9-gram	10-gram
Unique Tokens	15	209	1877	11408	51238	172109	441251	888837	1465215	2052063

documents. We perform a 5-fold Cross Validation and randomly down-sample non-sensitive documents to balance the training data.

Baseline: As a baseline we use a text classification approach with binary bag-of-words features. For this approach, we use SVM with a linear kernel and $C = 1.0$, since these *default* parameters are known to be effective for text classification [7, 14] and can provide a strong baseline for sensitivity classification [1, 13].

Sequence Classification: For the POS sequence representations, following [11, 12], we use the TreeTagger[1] part-of-speech tagger to POS tag documents using a reduced set of 15 POS tags. We then create separate n-gram sequence representations of the collection, resulting in individual n-gram sequence collections for $n = \{1...10\}$. Table 1 presents the number of observed unique tokens in the alphabet, A, for each size of n. For the linear and Gaussian kernels, we represent documents as token frequency vectors. For the Spectrum, Mismatch and Smith-Waterman kernels, we count the frequency of k length sub-sequence matches in a pair of documents. We train a separate committee classifier for each size of n-gram sequence, resulting in n *votes* per kernel as input to the ensemble approaches.

Ensemble Classification: For the ensemble approaches, we combine the predictions of the text classification P_t with the predictions of n sequence classifiers P_s, resulting in $n + 1$ document features $f, f \in \{p_t, p_s\}$. We test four combination methods. Firstly, in Weighted Majority Vote (WMV), to predict a document's class, p_t is assigned a weight w for each fold and the document's overall prediction score is calculated as $\frac{(p_t \cdot w)+\sum_{i=1}^{n} p_{s\,i}}{n+1}$. The remaining three combination methods are *stacking* approaches. These require an intermediate step where P_t and P_s are predictions for a validation set for each of the 5-fold Cross Validation folds. P_t and P_s are then concatenated and the resulting $n + 1$ predictions (per document) are used to train the combiner. We test three classifiers as combiners, namely Logistic Regression (LR), SVM and Random Forests (RF).

Classification and Parameters: We use scikit-learn[2] and extend LibSVM[3] with the Spectrum, Mismatch and Smith-Waterman kernels. Parameter values for the sequence and combinator classifiers are selected by 10-fold Cross Validation on training and validation sets respectfully, for each of the 5-fold Cross Validation folds. We vary SVM's C parameter exponentially in the range [0.001,10000], and similarly for γ parameters in [0.0001,10]. For sequence classification, sub-sequences are varied for $k = \{3, 6, 9, 12\}$. For ensemble combinators: for WMV, we test $w = \{1..100\}$; for LR we select L1 as our loss function and vary C in the same range as for SVM; for RF, we test number of trees $t = \{100, 250, 500, 750, 1000\}$. We optimise for area under the Receiver Operating Characteristic curve (auROC).

Metrics: We select auROC as our main metric for measuring kernel effectiveness, since it is calculated over all decision thresholds for a classifier. Additionally, we report precision (P), True Positive Rate (TPR), True Negative Rate (TNR), F_1, F_2 and Balanced Accuracy (BAC) metrics. We report statistical significance using McNemar's non-parametric test, with $p < 0.001$. Significant improvements compared to the text classification baseline are denoted by † in Table 4.

[1]http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/
[2]http://scikit-learn.org/
[3]https://csie.ntu.edu.tw/~cjlin/libsvm/

Table 2: Results for the best performing size of n-gram for *stand-alone* POS sequence classification, according to the area under the ROC curve (auROC).

	N	P	TPR	TNR	F_1	F_2	BAC	auROC
Linear	5	0.2185	0.6155	0.6651	0.3225	0.4514	0.6403	0.6897
Gaussian	4	0.2070	0.6494	0.6214	0.3139	**0.4550**	0.6354	0.6820
Spectrum	1	0.1868	0.6574	0.5644	0.2909	0.4370	0.6109	0.6636
Mismatch	1	0.1847	0.4833	0.6006	0.2673	0.3387	0.5420	0.5415
Smith-Waterman	2	0.2266	0.6250	0.6006	**0.3326**	0.3024	0.6128	0.6476
Hybrid								
Spectrum+Linear	4	0.2266	0.6178	0.6656	0.3278	0.4384	**0.6417**	0.6779
Spectrum+Gaussian	2	0.2245	0.5995	0.6780	0.3251	0.4361	0.6388	0.6764
Boosted								
Linear	1-6	**0.3433**	0.4074	**0.8087**	0.2874	0.3263	0.6081	0.7027
Gaussian	1-9	0.2290	0.5726	0.6511	0.2981	0.3933	0.6118	**0.7031**
Spectrum	1-3	0.1834	**0.7146**	0.4678	0.2829	0.4207	0.5912	0.6801

5 RESULTS

In this section, we first review the performance of each of the kernels as *stand-alone* classifiers for POS sequence classification without text features, before evaluating the combined *ensemble* approaches, compared to the text classification baseline.

Table 2 presents the results for the *stand-alone* classifiers. The table shows the best performing size of n-gram for each of the individual kernels and for two hybrid classifiers, namely *Spectrum+Linear* and *Spectrum+Gaussian*, according to auROC. Additionally, Table 2 also presents the results of a simple boosting classification approach where, for a specific kernel, we add the output from an n-gram classification as an additional feature for the n+1-gram classification.

As shown in Table 2, the linear kernel achieves the best auROC score (0.6897). However, the Gaussian and Spectrum kernels perform competitively with the linear kernel, achieving 0.6820 and 0.6636 auROC respectively. Moreover, in sensitivity classification the cost of mis-classifying a sensitive document is far greater than that of mis-classifying a not-sensitive document and the highest F_2 (0.4550) and TPR (0.6574) scores are achieved by the Gaussian and Spectrum kernels respectively. The Mismatch and Smith-Waterman kernels perform less well, achieving 0.5415 and 0.6476 auROC respectively. Therefore, in the remaining approaches, we focus on the Spectrum, Gaussian and linear kernels.

When evaluating the effectiveness of kernels, we are interested in notable differences in the correctness of predictions for sensitive documents. As shown in Table 3, there is substantial Fleiss' κ agreement between the linear and Gaussian kernels, but only moderate agreement between the Spectrum and linear or Gaussian kernels. This is in line with our expectation that sequence-based kernels, such as String kernels, can identify different features of sensitivity than vector space kernels, such as linear or Gaussian. Therefore, we select the Spectrum kernel as our base kernel for hybrid kernels.

As can be seen from Table 2, the hybrid kernels achieve 0.67 auROC. This is slightly less than the 0.68 auROC achieved by the linear and Gaussian kernels individually. However, in terms of balanced accuracy, the hybrid kernels improve overall performance (0.6417

Table 3: Fleiss' κ agreement between the linear, Gaussian and Spectrum kernels for predictions on sensitive documents, i.e. True Positive or False Negative predictions.

	Lin-Gau-Spec	Lin-Gau	Lin-Spec	Gau-Spec
Fleiss' κ	0.5312	0.7301	0.4502	0.4122

Table 4: Results for *ensemble* classification and TC baseline.

	# Votes		P	TPR	TNR	F_1	F_2	BAC	auROC
Text Classification (TC)			0.2410	0.6573	0.6841	0.3520	0.4874	0.6707	0.7419
Weighted Majority Vote (WMV)									
TC+POS$_{\text{Linear}}$	11	†	0.2610	0.6853	0.7048	0.3780	**0.5171**	0.6950	**0.7659**
TC+POS$_{\text{Gaussian}}$	11	†	0.2631	0.6813	0.7096	**0.3796**	0.5169	**0.6954**	0.7633
TC+POS$_{\text{Spectrum}}$	11	†	0.2412	**0.6932**	0.6681	0.3578	0.5042	0.6807	0.7588
TC+POS$_{\text{Lin,Gaus,Spec}}$	31	†	0.2578	0.6554	0.7129	0.3701	0.5009	0.6842	0.7616
TC+POS$_{\text{Spectrum+Linear}}$	11		0.2211	0.6295	0.6626	0.3273	0.4597	0.6461	0.7033
TC+POS$_{\text{Boosted Gaussian}}$	11		0.2445	0.6414	0.6984	0.3540	0.4842	0.6699	0.7308
logistic Regression (LR)									
TC+POS$_{\text{Linear}}$	11	†	0.2505	0.6752	0.6923	0.3646	0.5028	0.6837	0.7584
TC+POS$_{\text{Gaussian}}$	11	†	0.2437	0.6513	0.6923	0.3537	0.4865	0.6718	0.7492
TC+POS$_{\text{Spectrum}}$	11	†	0.2364	0.6495	0.6805	0.3462	0.4805	0.6650	0.7502
TC+POS$_{\text{Lin,Gaus,Spec}}$	31	†	0.2447	0.6594	0.6905	0.3559	0.4908	0.6749	0.7531
TC+POS$_{\text{Spectrum+Linear}}$	11	†	0.2451	0.6733	0.6845	0.3587	0.4978	0.6789	0.7502
TC+POS$_{\text{Boosted Gaussian}}$	11		0.2440	0.6294	0.7023	0.3507	0.4768	0.6659	0.7352
Support Vector Machine (SVM)									
TC+POS$_{\text{Linear}}$	11	†	0.2461	0.6695	0.6875	0.3589	0.4964	0.6785	0.7506
TC+POS$_{\text{Gaussian}}$	11		0.2385	0.6235	0.6963	0.3436	0.4691	0.6599	0.7398
TC+POS$_{\text{Spectrum}}$	11		0.2410	0.6256	0.6990	0.3463	0.4717	0.6623	0.7385
TC+POS$_{\text{Lin,Gaus,Spec}}$	31		0.2435	0.6236	0.7026	0.3488	0.4730	0.6631	0.7307
TC+POS$_{\text{Spectrum+Linear}}$	11		0.2455	0.6335	0.7011	0.3519	0.4782	0.6673	0.7452
TC+POS$_{\text{Boosted Gaussian}}$	11		0.2462	0.6215	0.7093	0.3515	0.4745	0.6654	0.7358
Random Forest (RF)									
TC + POS$_{\text{Linear}}$	11		**0.3858**	0.2629	0.9357	0.3091	0.2791	0.5993	0.7124
TC + POS$_{\text{Gaussian}}$	11		0.3531	0.2250	0.9363	0.2715	0.2412	0.5807	0.6975
TC + POS$_{\text{Spectrum}}$	11		0.3190	0.2349	0.9212	0.2672	0.2463	0.5780	0.6697
TC + POS$_{\text{Lin,Gaus,Spec}}$	31		0.3557	0.2230	**0.9369**	0.2718	0.2400	0.5800	0.6888
TC + POS$_{\text{Spectrum+Linear}}$	11		0.3522	0.2429	0.9321	0.2860	0.2583	0.5875	0.6974
TC + POS$_{\text{Boosted Gaussian}}$	11		0.3342	0.2407	0.9276	0.2775	0.2539	0.5842	0.7381

Spectrum+Linear vs. 0.6109 Spectrum and 0.6403 Linear, 0.6388 Spectrum+Gaussian vs. 0.6109 Spectrum and 0.6354 Gaussian).

The boosted classification approach, presented in Table 2, markedly improves auROC for the linear and Gaussian kernels (Linear 0.7027 vs. 0.6897, Gaussian 0.7031 vs. 0.6820). For these kernels, boosting notably increases precision (0.2290 vs. 0.2070 Gaussian, 0.3433 vs. 0.2185 Linear). The boosted Gaussian achieves the best auROC for stand-alone sequence classifiers (0.7031).

Moving on to *ensemble* classification, Table 4 presents the results for the four ensemble combination approaches WMV, LR, SVM, and RF. For each approach, the table presents the individual kernels (separately and together) and the best performing hybrid and boosted kernels, along with the text classification baseline.

Firstly, we note that text classification achieves 0.6707 BAC, markedly better than random (0.5 BAC), and 0.7419 auROC. Notably, text classification also performs better than the *stand-alone* classifiers from Table 2 (however, the stand-alone boosted approaches are competitive and achieve higher TPR and TNR scores).

Reviewing Table 4, we conclude that the linear kernel performs best for ensemble approaches, since it achieves significant improvements (denoted as †), and performs better for all measures, compared to the text classification baseline, when either of the WMV, LR or SVM combinators are deployed. This is surprising, since the hybrid and boosted kernels perform best for stand-alone classifiers. This appears be due to the liner kernel model being more similar to the (better) text classification model than the other kernel models are, while having enough uncorrelated variations to enhance the predictions. We will investigate this further as future work.

Turning our attention to the combinator methods, we see that LR achieves significant improvements compared to text classification for five of the six kernel combinations tested (+1.2-2.3% auROC) and is clearly the most effective stacked ensemble approach. However, we conclude that WMV performs better than the stacked approaches since it achieves the highest TPR, F_1, F_2, BAC and auROC scores. Currently, WMV applies $p_t \cdot w$ globally and we expect to be able to further improve these results by learning more fine grained weights, at the instance or vote level. Again, we leave this as future work.

Overall, combining text classification with linear kernel POS sequence classification (TC+POS$_{\text{Linear}}$) and WMV performs best for sensitivity classification, from the combinations we tested. This approach achieves significant improvements according to McNemar's test with $p < 0.001$ (+6.09% F_2, +3.24% auROC), compared to the text classification baseline. Moreover, the approach correctly predicted 4.25% more sensitive documents (0.6853 TPR vs. 0.6573 TPR), while achieving a 3.02% increase in correct not-sensitive predictions (0.7048 TNR vs. 0.6841 TNR). This results in an additional 83 correct predictions on our collection.

These results show that combining text classification with POS sequence classification can be effective for classifying documents that contain sensitivities relating to FOI exemptions. Moreover, the largest gains in overall classification performance can be achieved when deploying a simple weighted majority vote combination strategy and a linear SVM kernel for POS sequence classification. This, in turn, has the additional advantage of reducing training times.

6 CONCLUSIONS

In this work, we presented a thorough investigation of five SVM kernel functions (Linear, Gaussian, Spectrum, Mismatch and Smith-Waterman) for sensitivity classification using Part-of-Speech n-gram sequences. We showed that an ensemble classification approach that combines text classification with sequence classification can significantly improve sensitivity classification effectiveness. Moreover, we found that combining linear kernel POS sequence classification with text classification by Weighted Majority Vote lead to the largest increase in sensitivity classification effectiveness (+ 6.09% F_2), when compared to a text classification baseline.

REFERENCES

[1] Giacomo Berardi, Andrea Esuli, Craig Macdonald, Iadh Ounis, and Fabrizio Sebastiani. 2015. Semi-automated text classification for sensitivity identification. In *Proc. CIKM.*

[2] Bee-Chung Chen, Daniel Kifer, Kristen LeFevre, Ashwin Machanavajjhala, and others. 2009. Privacy-preserving data publishing. *Foundations and Trends® in Databases* 2, 1–2 (2009).

[3] TG Ditterrich. 1997. Machine learning research: four current directions. *Artificial Intelligence Magzine* 4 (1997).

[4] Eleazar Eskin, Jason Weston, William S Noble, and Christina S Leslie. 2002. Mismatch string kernels for SVM protein classification. In *Proc. NIPS.*

[5] Aris Gkoulalas-Divanis, Grigorios Loukides, and Jimeng Sun. 2014. Publishing data from electronic health records while preserving privacy. *Journal of Biomedical Informatics* 50 (2014).

[6] Timothy Gollins, Graham McDonald, Craig Macdonald, and Iadh Ounis. 2014. On using information retrieval for the selection and sensitivity review of digital public records. In *Proc. PIR.*

[7] Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proc. ECML.*

[8] S Sathiya Keerthi and Chih-Jen Lin. 2003. Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural computation* 15, 7 (2003).

[9] Ludmila I Kuncheva and Juan J Rodríguez. 2014. A weighted voting framework for classifiers ensembles. *Knowledge and Information Systems* 38, 2 (2014).

[10] Christina S Leslie, Eleazar Eskin, and William Stafford Noble. 2002. The spectrum kernel: A string kernel for SVM protein classification. In *Proc. PSB.*

[11] C. Lioma and I. Ounis. 2006. Examining the content load of part-of-speech blocks for information retrieval. In *Proc. COLING/ACL.*

[12] Graham McDonald, Craig Macdonald, and Iadh Ounis. 2015. Using part-of-speech n-grams for sensitive-text classification. In *Proc. ICTIR.*

[13] Graham McDonald, Craig Macdonald, Iadh Ounis, and Timothy Gollins. 2014. Towards a classifier for digital sensitivity review. In *Proc. ECIR.*

[14] Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys* 34, 1 (2002).

[15] Temple F Smith and Michael S Waterman. 1981. Identification of common molecular subsequences. *Journal of molecular biology* 147, 1 (1981).

[16] Vladimir Naumovich Vapnik and Vlamimir Vapnik. 1998. *Statistical learning theory.* Vol. 1. Wiley New York.

[17] David H Wolpert. 1992. Stacked generalization. *Neural networks* 5, 2 (1992).

[18] Zhengzheng Xing, Jian Pei, and Eamonn Keogh. 2010. A brief survey on sequence classification. *ACM SIGKDD Explorations Newsletter* 12, 1 (2010).