Project

Re-implementation and analysis of the "Text Classification using String Kernels" paper, by Lodhi, Saunders, Shawe-Taylor, Cristianini and Watkins [3].

Group 13
V. Polianskii, B. Godefroy, W. Kryscinski, F. Franzen

KTH Royal Institute of Technology
Machine Learning, advanced course

Description of the article

The paper by Joachims [2] has shown that Support Vector Machine (SVM) was a good candidate for text categorization, since this algorithm can successfully handle some very high dimensional sparse features vectors.

Then, there is a need to transform documents (strings of characters) into a suitable representation for SVM, and this is performed by a kernel, a function that maps two input strings into a feature space, i.e. a vector. For this task, some techniques already exist, but the article [3] proposes a new one, called string subsequence kernel (SSK), which could outperform previous text categorization performances. Since its computational cost make it almost unusable in practice for big datasets, the paper also provides an estimation technique of this new kernel.

First of all, all standard text representation techniques remove non-informative words (stop words, such as conjunctions) and replace words by their stems, assuming that this information loss is of minor importance. The new approach proposed does the same. Then, [2] has proposed a standard word feature space kernel (WK), using the text representation technique [1], which maps the input string to a high dimensional feature vector, only considering the words frequency. This approach is efficient and provides good results but it loses the word order information. The n-grams kernel (NGK) uses a similar principle, but considers n-grams (contiguous sequences of n letters) frequency instead of the words frequency. Then, the idea behind the new kernel SSK is to use the frequency of non-contiguous sequences of n symbols, in other words, to capture more information from the input documents than the previous techniques.

The purpose of the new kernel is to compare the similarity between two documents, assuming that the more substrings they have in common, the more similar they are. Since SSK deals with non-contiguous sequences, it introduces a decay factor, $\lambda \in (0;1)$, which performs a weighting according to the sequences lengths.

Using the example from [3], let's consider 4 simple documents, each one composed of a single word for simplicity: *cat*, *car*, *bat* and *bar*. With a subsequence length $k = 2$, we obtain a 8-dimensional feature space, as follows:

	c-a	c-t	a-t	b-a	b-t	c-r	a-r	b-r
$\phi(cat)$	λ^2	λ^3	λ^2	0	0	0	0	0
$\phi(car)$	λ^2	0	0	0	0	λ^3	λ^2	0
$\phi(bat)$	0	0	λ^2	λ^2	λ^3	0	0	0
$\phi(bar)$	0	0	0	λ^2	0	0	λ^2	λ^3

Then, we can compute the unnormalised kernel between 2 documents by performing

a scalar product between their feature vectors. For instance,

$$K(cat, car) = \lambda^4$$
$$K(car, bat) = 0$$

After normalisation, we obtain,

$$K(cat, car) = \frac{\lambda^4}{K(cat, cat)} \ (= \frac{\lambda^4}{K(car, car)}) \ = \frac{\lambda^4}{2\lambda^4 + \lambda^6} = \frac{1}{2 + \lambda^2}$$
$$K(car, bat) = 0$$

Using this new kernel, we can compute the similarity between any documents. For instance, considering

$$K(\text{"science is organized knowledge"}, \text{"wisdom is organized life"})$$

we obtain, for $k = 1, 2, 3, 4, 5, 6$, $K_1 = 0.580$, $K_2 = 0.580$, $K_3 = 0.478$, $K_4 = 0.439$, $K_5 = 0.406$ and $K_6 = 0.370$ [3].

The direct computation of the SSK would involve $O(|\Sigma|^n)$ time and space, with Σ the alphabet used by the documents and n the subsequences length [3]. Unfortunately, this computation is unpractical, even for normal sized documents and small values of n. That's why the article proposes a dynamic programming technique that makes it a lot more efficient, and reduces its complexity to $O(n|s||t|)$, with n the length of the subsequences, and s, t the documents we compare using the kernel.

However, this is still too expensive for the Reuters dataset, which contains approximately 9600 training examples and 3200 test examples with an average length of approximately 2300 characters. That's why the article proposes an estimate computation of SSK. It consists in selecting the substrings that occur the most frequently in the dataset, i.e. which are more likely to be highly informative, and this reduces the feature space size and the number of kernel evaluations. According to the tests performed in the article, the alignment measure [4] shows that the deviation of the approximate Gram matrix from the true SSK Gram matrix is very little, and consequently, the approximate SSK should provide similar results to the standard SSK technique while having a lowest computational cost.

Re-implementation

In this project, we have re-implemented the SSK method and its approximated version as described in the original paper. For this purpose we used Python with additional, third-party libraries. We chose to focus on implementing the kernel functions that measure similarity between documents and used the SVM algorithm implementation offered by the scikit-learn library. For the performance evaluation, we used the same measures as in the paper: precision, recall and F1-score.

The first step in the project was to preprocess the Reuters dataset as described by the authors of the paper. We used the NLTK library for this purpose. After removing popular stop-words and punctuation we split the data into a training and a test dataset.

We started with implementing the direct (naive) algorithm, but as stated in the paper, its exponential complexity makes it infeasible to use on documents longer than a couple dozen characters. We used the naive implementation as "ground truth" when (internally) testing the more sophisticated version of the algorithm.

The next step was to implement the recursive version of the algorithm and to add some efficiency upgrades as suggested by the authors. Although the recursive version of the algorithm was several orders of magnitude faster than the naive implementation, it remained too slow to compute the similarity between documents in the dataset. To limit the amount of recomputation while performing tests we decided to precompute the Gram matrices. The Gram matrix of the training dataset was 380×380, while the test dataset was 380×90. To make the similarity computations faster, we moved the crucial parts of the code to Cython and added parallel computation, using Python multiprocessing, on top of that.

Based on experiments and experience, we estimated that the time to precompute a pair of Gram matrices (training and test) for a single experiment would take between 1 and 8 hours depending on the settings of the SSK algorithm. The computation time heavily depended on the length n of the subsequence. Since we had to precompute a total of 38 Gram matrices (19 different n and λ settings in SSK) we chose to use compute optimized instances offered by AWS. Using a 32-core 8x.large instance we were able to precompute the Gram matrices in 30 to 75 minutes, again depending on the SSK algorithm settings.

The experiments ran by the authors of the paper were repeated 10 times and the results were averaged. Unfortunately we were not able to follow this practice due to the long computation time, for this project we ran 1 experiment per SSK setting. As in the original paper experiments were conducted on documents in four categories: "earn", "acq", "crude", and "corn".

Then, an approximated version of SSK was implemented and evaluated on the Reuters data set. The computation process of a Gram matrix for an approximated kernel was divided into three parts: extraction of substring set $\tilde{S}$ from a training data set, transformation of data sets into $|\tilde{S}|$-dimensional space by computing kernel values between each document and each substring, computation of Gram matrix using Euclidean distance between obtained points to estimate the similarity between them.

In order to compare techniques performances, we have also re-implemented WK and NGK kernels.

Results

Experiment results show the effectiveness of varying both the subsequence length and the weight decay factor of the SSK kernel. To make the effects easily interpretable only one paramters was varied in each experiment and the other was held constant. In both cases results are contrasted with those obtained using other kernel functions (WK and NGK). As previously mentioned, for each parameter setting the experiment was performed once and the train and test dataset was the same across experiments.

In the case of the subsequence length, the decay value was set to $\lambda = 0.5$ and lengths $n = 3, ..., 14$. were tested. Partial results for two largest categories are presented in table 1.

Category	**Kernel**	**Length** n	**Original article**		**Re-implementation**		
			F1		**F1**	**Precision**	**Recall**
			Mean	SD	Mean	Mean	Mean
earn	**SSK**	3	0.925	0.036	0.974	1.000	0.950
		7	**0.940**	0.035	**1.000**	1.000	1.000
		14	0.936	0.027	0.656	0.488	1.000
	NGK	3	0.919	0.035	0.987	1.000	0.975
		7	**0.940**	0.035	0.947	1.000	0.900
		14	0.923	0.034	0.333	1.000	0.200
	WK		0.925	0.033	0.988	0.976	1.000
Category	**Kernel**	**Length** n	**Original article**		**Re-implementation**		
			F1		**F1**	**Precision**	**Recall**
			Mean	SD	Mean	Mean	Mean
acq	**SSK**	3	0.785	0.040	**0.909**	0.833	1.000
		7	0.864	0.045	0.893	0.806	1.000
		14	0.774	0.042	0.303	0.625	0.200
	NGK	3	0.791	0.043	0.833	0.869	0.800
		7	**0.870**	0.050	0.666	0.823	0.560
		14	0.776	0.060	0.068	0.250	0.04
	WK		0.802	0.072	0.902	0.885	0.920

Table 1: Partial results from experiment reproduction. The table shows how classification performance varies depending on the SSK subsequence length n. Full results are available in the appendix.

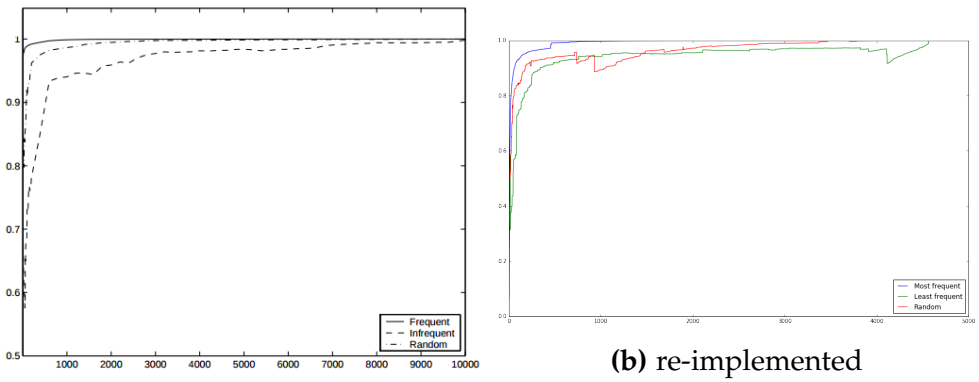
In the case of the weight decay factor, the subsequence length was set to $n = 5$ and decays $\lambda = 0.01, ..., 0.9$. were tested. Partial results for two largest categories are presented in table 2.

Category	Kernel	Decay λ	Original article		Re-implementation		
			F1		F1	Precision	Recall
			Mean	SD	Mean	Mean	Mean
earn	**SSK**	0.01	**0.946**	0.026	**1.000**	1.000	1.000
		0.1	0.944	0.026	**1.000**	1.000	1.000
		0.9	0.914	0.050	0.949	0.974	0.925
	NGK	0	0.925	0.014	0.961	1.000	0.925
	WK	0	0.882	0.038	0.974	1.000	0.950
Category	**Kernel**	**Decay λ**	**Original article**		**Re-implementation**		
			F1		F1	Precision	Recall
			Mean	SD	Mean	Mean	Mean
acq	**SSK**	0.01	**0.943**	0.893	**1.000**	1.000	1.000
		0.1	0.871	0.043	0.943	0.893	1.000
		0.9	0.735	0.073	0.745	0.731	0.760
	NGK	0	0.882	0.038	0.833	0.869	0.800
	WK	0	0.802	0.033	0.902	0.885	0.920

Table 2: Partial results from experiment reproduction. The table shows how classification performance varies depending on the SSK decay λ. Full results are available in the appendix.

For the approximated kernel, firstly, the alignment measure between a true kernel and an approximated kernel was measured while varying the substring set size. The alignment measure is obtained as a normalized Frobenius inner product between estimated Gram matrices.

For this experiment only first 100 documents were used. The plots for this comparison are shown on figure 1 (the parameters for kernels are: $n = 5$ and $\lambda = 0.5$). *Frequent*, *Infrequent* and *Random* denote how the substrings were selected to form the subset - either those which appear more frequently in the training data set, or less frequently, or randomly.



(a) original

(b) re-implemented

Figure 1: Comparison of alignment scores against the Gram matrix generated by the full string kernel.

Then, SVM with approximated kernel was tested on the whole data set. Performance with two sizes of substring sets were compared - 1000 and 3000, the results are shown in the table 3. $k = 5$ was used as the parameter.

	Original		Implemented	
	1000	3000	1000	3000
earn	0.97	0.97	0.97	0.97
acq	0.88	0.85	0.85	0.87
ship	0.10	0.53	0.55	0.57
corn	0.15	0.65	0.00	0.50

Table 3: The table shows F1-scores for 4 categories with different numbers of features (1000 and 3000) in kernel approximation.

Lastly, F1-scores were compared for different k parameter for top-10 categories in the data set. The results are in the table 4.

Category	Original approx. SSK			Implemented approx. SSK		
	$k = 3$	$k = 4$	$k = 5$	$k = 3$	$k = 4$	$k = 5$
earn	0.970	0.970	0.970	0.964	0.967	0.967
acq	0.850	0.880	0.880	0.850	0.872	0.873
money-fx	0.700	0.760	0.740	0.621	0.693	0.703
grain	0.800	0.820	0.800	0.722	0.768	0.766
crude	0.820	0.840	0.840	0.707	0.728	0.736
trade	0.700	0.730	0.730	0.691	0.766	0.754
interest	0.630	0.660	0.690	0.561	0.616	0.620
ship	0.610	0.650	0.530	0.589	0.560	0.571
wheat	0.780	0.790	0.820	0.741	0.767	0.743
corn	0.630	0.630	0.680	0.328	0.486	0.500

Table 4: The table shows F1-scores of the approximated SSK on top-5 categories of complete dataset.

Discussion

Results of experiments where the subsequence length n was varied (table 1 and tables 5, 6 in the appendix) show that the SSK has better performance on subsequences of short-mid length. Across categories the performance measured with the F1-score seemed to peak for lengths between 4 and 7. For larger values of n we observed decreased performance of the algorithm. These results follow those obtained by the authors of the paper. When comparing the SSK kernel with the NGK we first notice that the NGK has the best performance for small values of n. In all cases but one it reaches peak performance for $n = 3$ and in the outlying case $n = 4$. From the experiment results we see that the performance of both kernels does not drastically differ (specially for low n) but in general SSK slightly outperforms the n-gram kernel. It is

also interesting to compare the word kernel with the SSK, which performs relatively well and in some categories ("crude" and "corn") outperforms the SSK.

Results of experiments where the weight decay factor λ was varied (table 2 and tables 7, 8 in the appendix) show that the performance of the SSK isn't very sensitive to changes in the λ parameter. For all categories we see that the F1-score (within that category) was very similar or even constant for decay values from 0.01 to 0.5. The more extreme cases of 0.5 and 0.7 usually significantly decrease classification performance. The SSK kernel outperformed both the NGK and WK in all categories except for "crude" where the WK had a higher F1-score.

In general, the results obtained from our experiments with the SSK kernel are similar to those presented in the paper. The performance metrics (F1, precision and recall) values are close, but not exactly the same as the ones presented by the authors. One possible reason for this is that we ran each experiment only once, due to mentioned computational limitations, and in the paper each experiment was repeated 10 times on a differently sampled test set. The remarkably good (e.g. F1-score of 1.0) and bad (e.g. F1-score of 0.0) performance in some categories can be explained by some bias carried by the small sampled test dataset. It is possible that the sampled documents were relatively easy to categorize, which would explain the high values of performance metrics. It is also possible that these documents had the tendency to be short in length, which would explain the bad performance for longer subsequences.

Testing of the approximated kernel gives results very similar to the original numbers and the conclusions might be the same for that part as in the main article. The reason for that is that the data sets used in the original article and in the re-implementation are the same, as well as the sets of substrings (but not absolutely the same because several factor might differ slightly, for example preprocessing of the documents). Interesting to note that there was a problem of low F1-score of the categories "ship" and "corn" with $|\tilde{S}| = 1000$ in the article, though this problem didn't appear for "ship" while testing.

After having reproduced and analyzed the results obtained in the paper, we can say that this technique proved to be a very interesting and performant approach for text categorization. Unfortunately, the huge amount of computation it requires makes it hard to use with datasets larger than a couple of hundred documents, which in today scale can be called very small.

References

[1] A. Wong G. Salton and C. Yang. *A Vector Space Model for Automatic Indexing*. Communications of the ACM, 18(11):613–620. 1975.

[2] T. Joachims. *Text categorization with support vector machines: Learning with many relevant features*. Claire Nédellec and Céline Rouveirol, editors, Proceedings of the European Conference on Machine Learning, pages 137–142, Berlin, 1998. Springer. 1998.

[3] Shawe-Taylor Cristianini Lodhi Saunders and Watkins. *Text Classification using String Kernels*. Mach. Learn. Res. 2, pp. 419-444. 2002.

[4] J. Shawe-Taylor N. Cristitiaini A. Elissee and J. Kandola. *On kernel-target alignment*. NIPS '01. 2002.

A Full experiment reproduction results

Category	Kernel	Length n	Original article		Re-implementation		
			F1		F1	Precision	Recall
			Mean	SD	Mean	Mean	Mean
earn	**SSK**	3	0.925	0.036	0.974	1.000	0.950
		4	0.932	0.029	0.988	0.976	1.000
		5	0.936	0.036	**1.000**	1.000	1.000
		6	0.936	0.033	**1.000**	1.000	1.000
		7	0.940	0.035	**1.000**	1.000	1.000
		8	0.934	0.033	**1.000**	1.000	1.000
		10	0.927	0.032	0.964	0.930	1.000
		12	0.931	0.036	0.734	0.580	1.000
		14	0.936	0.027	0.656	0.488	1.000
	NGK	3	0.919	0.035	0.987	1.000	0.975
		4	0.943	0.030	0.961	1.000	0.900
		5	**0.944**	0.026	0.961	1.000	0.925
		6	0.943	0.030	0.961	1.000	0.925
		7	0.940	0.035	0.947	1.000	0.900
		8	0.940	0.045	0.933	1.000	0.800
		10	0.932	0.032	0.888	1.000	0.750
		12	0.917	0.033	0.596	1.000	0.435
		14	0.923	0.034	0.333	1.000	0.200
	WK		0.925	0.033	0.988	0.976	1.000
Category	**Kernel**	**Length n**	**Original article**		**Re-implementation**		
			F1		**F1**	**Precision**	**Recall**
			Mean	SD	Mean	Mean	Mean
acq	**SSK**	3	0.785	0.040	0.909	0.833	1.000
		4	0.822	0.047	0.923	0.889	0.960
		5	0.867	0.038	**0.943**	0.893	1.000
		6	0.876	0.051	**0.943**	0.893	1.000
		7	0.864	0.045	0.893	0.806	1.000
		8	0.852	0.049	0.893	0.806	1.000
		10	0.791	0.67	0.774	0.649	0.960
		12	0.791	0.067	0.696	0.762	0.640
		14	0.774	0.042	0.303	0.625	0.200
	NGK	3	0.791	0.043	0.833	0.869	0.800
		4	0.873	0.031	0.857	0.875	0.84
		5	**0.882**	0.038	0.833	0.869	0.800
		6	0.880	0.045	0.808	0.863	0.760
		7	0.870	0.050	0.666	0.823	0.560
		8	0.857	0.044	0.634	0.812	0.520
		10	0.830	0.045	0.193	0.500	0.120
		12	0.806	0.066	0.068	0.250	0.04
		14	0.776	0.060	0.068	0.250	0.04
	WK		0.802	0.072	0.902	0.885	0.920

Table 5: Full results from experiment reproduction. The table shows how classification performance varies depending on the SSK subsequence length n.

Category	Kernel	Length n	Original article		Re-implementation		
			F1		F1	Precision	Recall
			Mean	SD	Mean	Mean	Mean
crude	**SSK**	3	0.881	0.077	0.788	0.722	0.867
		4	0.905	0.090	0.839	0.812	0.867
		5	0.936	0.045	0.839	0.812	0.867
		6	0.901	0.051	0.812	0.765	0.867
		7	0.872	0.050	0.828	0.857	0.800
		8	0.828	0.066	0.828	0.857	0.800
		10	0.764	0.098	0.636	1.000	0.467
		12	0.709	0.111	0.000	0.000	0.000
		14	0.761	0.106	0.000	0.000	0.000
	NGK	3	0.907	0.060	0.857	0.923	0.8
		4	0.935	0.041	0.888	1.000	0.800
		5	**0.937**	0.048	0.888	1.000	0.800
		6	0.908	0.041	0.888	1.000	0.800
		7	0.904	0.054	0.846	1.000	0.733
		8	0.869	0.060	0.750	1.000	0.600
		10	0.811	0.090	0.333	1.000	0.200
		12	0.737	0.098	0.000	0.000	0.000
		14	0.884	0.171	0.000	0.000	0.000
	WK		0.904	0.043	**0.889**	1.000	0.800
Category	**Kernel**	**Length n**	**Original article**		**Re-implementation**		
			F1		**F1**	**Precision**	**Recall**
			Mean	SD	Mean	Mean	Mean
corn	**SSK**	3	0.665	0.169	0.571	1.000	0.400
		4	0.783	0.103	0.750	1.000	0.600
		5	0.779	0.104	0.750	1.000	0.600
		6	0.749	0.096	0.667	1.000	0.500
		7	0.643	0.107	0.667	1.000	0.500
		8	0.569	0.099	0.667	1.000	0.500
		10	0.582	0.107	0.462	1.000	0.300
		12	0.618	0.086	0.000	0.000	0.000
		14	0.702	0.114	0.000	0.000	0.000
	NGK	3	0.797	0.068	0.823	1.000	0.700
		4	0.841	0.071	0.823	1.000	0.700
		5	**0.847**	0.103	0.823	1.000	0.500
		6	0.815	0.089	0.666	1.000	0.500
		7	0.767	0.117	0.666	1.000	0.500
		8	0.706	0.125	0.571	1.000	0.400
		10	0.646	0.113	0.571	1.000	0.400
		12	0.675	0.131	0.461	1.000	0.300
		14	0.813	0.174	0.000	0.000	0.000
	WK		0.762	0.099	**0.824**	1.000	0.700

Table 6: Full results from experiment reproduction. The table shows how classification performance varies depending on the SSK subsequence length n.

Category	**Kernel**	**Decay λ**	**Original article**		**Re-implementation**		
			F1		**F1**	**Precision**	**Recall**
			Mean	SD	Mean	Mean	Mean
earn	**SSK**	0.01	0.946	0.026	1.000	1.000	1.000
		0.03	0.946	0.028	1.000	1.000	1.000
		0.05	0.944	0.028	1.000	1.000	1.000
		0.07	0.944	0.026	1.000	1.000	1.000
		0.09	0.944	0.026	1.000	1.000	1.000
		0.1	0.944	0.026	1.000	1.000	1.000
		0.3	0.943	0.029	1.000	1.000	1.000
		0.5	0.936	0.013	1.000	1.000	1.000
		0.7	0.928	0.040	0.987	1.000	0.975
		0.9	0.914	0.050	0.949	0.974	0.925
	NGK	0	0.925	0.014	0.961	1.000	0.925
	WK	0	**0.882**	0.038	0.988	0.976	1.000
Category	**Kernel**	**Decay λ**	**Original article**		**Re-implementation**		
			F1		**F1**	**Precision**	**Recall**
			Mean	SD	Mean	Mean	Mean
acq	**SSK**	0.01	0.943	0.893	1.000	1.000	1.000
		0.03	0.878	0.040	0.943	0.893	1.000
		0.05	**0.882**	0.037	0.943	0.893	1.000
		0.07	0.873	0.044	0.943	0.893	1.000
		0.09	0.863	0.043	0.943	0.893	1.000
		0.1	0.871	0.043	0.943	0.893	1.000
		0.3	0.870	0.040	0.943	0.893	1.000
		0.5	0.867	0.038	0.943	0.893	1.000
		0.7	0.805	0.050	0.923	0.889	0.960
		0.9	0.735	0.073	0.745	0.731	0.760
	NGK	0	0.882	0.038	0.833	0.869	0.800
	WK	0	0.802	0.033	0.902	0.885	0.920

Table 7: Full results from experiment reproduction. The table shows how classification performance varies depending on the SSK weight decay factor λ.

Category	Kernel	Decay λ	Original article		Re-implementation		
			F1		F1	Precision	Recall
			Mean	SD	Mean	Mean	Mean
crude	**SSK**	0.01	0.937	0.048	0.867	0.867	0.867
		0.03	0.941	0.041	0.867	0.867	0.867
		0.05	0.945	0.041	0.867	0.867	0.867
		0.07	0.945	0.041	0.867	0.867	0.867
		0.09	0.927	0.052	0.867	0.867	0.867
		0.1	0.947	0.039	0.867	0.867	0.867
		0.3	**0.948**	0.030	0.839	0.812	0.867
		0.5	0.936	0.045	0.839	0.812	0.867
		0.7	0.893	0.363	0.743	0.650	0.867
		0.9	0.758	0.861	0.595	0.500	0.733
	NGK	0	0.847	0.048	0.846	1.000	0.733
	WK	0	0.904	0.043	0.889	1.000	0.800
Category	**Kernel**	**Decay λ**	**Original article**		**Re-implementation**		
			F1		**F1**	**Precision**	**Recall**
			Mean	SD	Mean	Mean	Mean
corn	**SSK**	0.01	0.845	0.098	0.824	1.000	0.700
		0.03	0.845	0.098	0.824	1.000	0.700
		0.05	0.834	0.086	0.824	1.000	0.700
		0.07	0.827	0.088	0.824	1.000	0.700
		0.09	0.834	0.083	0.824	1.000	0.700
		0.1	0.827	0.088	0.824	1.000	0.700
		0.3	0.825	0.087	0.750	1.000	0.600
		0.5	0.779	0.104	0.750	1.000	0.600
		0.7	0.628	0.109	0.571	1.000	0.400
		0.9	0.348	0.185	0.571	1.000	0.400
	NGK	0	**0.847**	0.103	0.823	1.000	0.500
	WK	0	0.762	0.099	0.824	1.000	0.700

Table 8: Full results from experiment reproduction. The table shows how classification performance varies depending on the SSK weight decay factor λ.