Tópicos de Minería de Datos

Text Classification using String Kernels

Katherine Sullivan
FCEIA - UNR

Índice

1. ¿Por qué es importante la tarea de la clasificación de textos?
2. Un pequeño repaso sobre kernels
   - El "truquito" del kernel
   - Kernels tradicionales vs kernels sobre conjuntos generales
3. Sobre el Paper
   - ¿Qué había hasta el momento?
   - ¿Qué enfoque distinto plantea el nuevo kernel?
   - ¿Cómo se hace eficiente el cálculo?
   - Resultados
4. Conclusiones

Índice

1. ¿Por qué es importante la tarea de la clasificación de textos?
2. Un pequeño repaso sobre kernels
   - El "truquito" del kernel
   - Kernels tradicionales vs kernels sobre conjuntos generales
3. Sobre el Paper
   - ¿Qué había hasta el momento?
   - ¿Qué enfoque distinto plantea el nuevo kernel?
   - ¿Cómo se hace eficiente el cálculo?
   - Resultados
4. Conclusiones

¿Por qué es importante la tarea de la clasificación de textos?

¿Por qué es importante la tarea de la clasificación de textos?

- Visión pragmática sobre usos actuales

¿Por qué es importante la tarea de la clasificación de textos?

- Visión pragmática sobre usos actuales
- Visión idealista sobre lo que representa el conocer a través de texto

Índice

1 ¿Por qué es importante la tarea de la clasificación de textos?

2 Un pequeño repaso sobre kernels
- El "truquito" del kernel
- Kernels tradicionales vs kernels sobre conjuntos generales

3 Sobre el Paper
- ¿Qué había hasta el momento?
- ¿Qué enfoque distinto plantea el nuevo kernel?
- ¿Cómo se hace eficiente el cálculo?
- Resultados

4 Conclusiones

El truquito del kernel

El truquito del kernel

- En varios problemas se puede desear mapear los datos originales a un espacio de características de mayor dimensión para que sean más fácilmente separables o para capturar patrones más complejos.

El truquito del kernel

- En varios problemas se puede desear mapear los datos originales a un espacio de características de mayor dimensión para que sean más fácilmente separables o para capturar patrones más complejos.
- En lugar de calcular explícitamente las transformaciones de características y luego realizar el producto interno en el espacio de características de mayor dimensión, el "kernel trick" permite calcular el producto interno directamente en el espacio original.

El truquito del kernel

- En varios problemas se puede desear mapear los datos originales a un espacio de características de mayor dimensión para que sean más fácilmente separables o para capturar patrones más complejos.
- En lugar de calcular explícitamente las transformaciones de características y luego realizar el producto interno en el espacio de características de mayor dimensión, el "kernel trick" permite calcular el producto interno directamente en el espacio original.
- ¿Cómo?

El truquito del kernel

- En varios problemas se puede desear mapear los datos originales a un espacio de características de mayor dimensión para que sean más fácilmente separables o para capturar patrones más complejos.
- En lugar de calcular explícitamente las transformaciones de características y luego realizar el producto interno en el espacio de características de mayor dimensión, el "kernel trick" permite calcular el producto interno directamente en el espacio original.
- ¿Cómo? ¡A través de las funciones de kernel!

El truquito del kernel

- En varios problemas se puede desear mapear los datos originales a un espacio de características de mayor dimensión para que sean más fácilmente separables o para capturar patrones más complejos.
- En lugar de calcular explícitamente las transformaciones de características y luego realizar el producto interno en el espacio de características de mayor dimensión, el "kernel trick" permite calcular el producto interno directamente en el espacio original.
- ¿Cómo? ¡A través de las funciones de kernel!
- Formalmente, una función de kernel $K(x, y)$ cumple con la propiedad de que existe una transformación de características ϕ tal que $K(x, y) = <\phi(x), \phi(y)>$

Kernels tradicionales vs kernels sobre conjuntos generales

Kernels tradicionales vs kernels sobre conjuntos generales

- ¿Cómo construimos un kernel?

Kernels tradicionales vs kernels sobre conjuntos generales

- ¿Cómo construimos un kernel?
- Kernels tradicionales suelen hacer uso del teorema de Mercer, que asegura que si K(x, y) es una función simétrica positiva definida cumple la propiedad de ser el producto interno de dos transformaciones.

Kernels tradicionales vs kernels sobre conjuntos generales

- ¿Cómo construimos un kernel?
- Kernels tradicionales suelen hacer uso del teorema de Mercer, que asegura que si K(x, y) es una función simétrica positiva definida cumple la propiedad de ser el producto interno de dos transformaciones.
- ¡Pero existe otra manera!

Kernels tradicionales vs kernels sobre conjuntos generales

- ¿Cómo construimos un kernel?
- Kernels tradicionales suelen hacer uso del teorema de Mercer, que asegura que si K(x, y) es una función simétrica positiva definida cumple la propiedad de ser el producto interno de dos transformaciones.
- ¡Pero existe otra manera! Definimos el kernel directamente como un producto interno.

Kernels tradicionales vs kernels sobre conjuntos generales

- ¿Cómo construimos un kernel?
- Kernels tradicionales suelen hacer uso del teorema de Mercer, que asegura que si K(x, y) es una función simétrica positiva definida cumple la propiedad de ser el producto interno de dos transformaciones.
- ¡Pero existe otra manera! Definimos el kernel directamente como un producto interno.
- Este es generalmente el abordaje que se usa para definir kernels sobre conjuntos generales.

Índice

1 ¿Por qué es importante la tarea de la clasificación de textos?

2 Un pequeño repaso sobre kernels
- El "truquito" del kernel
- Kernels tradicionales vs kernels sobre conjuntos generales

3 Sobre el Paper
- ¿Qué había hasta el momento?
- ¿Qué enfoque distinto plantea el nuevo kernel?
- ¿Cómo se hace eficiente el cálculo?
- Resultados

4 Conclusiones

¿Qué había hasta el momento?

¿Qué había hasta el momento?

- Bag of words.

¿Qué había hasta el momento?

- Bag of words.
- N-gram kernels.

¿Qué enfoque distinto plantea el nuevo kernel?

¿Qué enfoque distinto plantea el nuevo kernel?

Definition 1 (String subsequence kernel- SSK) *Let Σ be a finite alphabet. A string is a finite sequence of characters from Σ, including the empty sequence. For strings s, t, we denote by $|s|$ the length of the string $s = s_1 \ldots s_{|s|}$, and by st the string obtained by concatenating the strings s and t. The string $s[i:j]$ is the substring $s_i \ldots s_j$ of s. We say that u is a subsequence of s, if there exist indices $\mathbf{i} = (i_1, \ldots, i_{|u|})$, with $1 \leq i_1 < \cdots < i_{|u|} \leq |s|$, such that $u_j = s_{i_j}$, for $j = 1, \ldots, |u|$, or $u = s[\mathbf{i}]$ for short. The length $l(\mathbf{i})$ of the subsequence in s is $i_{|u|} - i_1 + 1$. We denote by Σ^n the set of all finite strings of length n, and by Σ^* the set of all strings*

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n. \tag{1}$$

We now define feature spaces $F_n = \mathbb{R}^{\Sigma^n}$. The feature mapping ϕ for a string s is given by defining the u coordinate $\phi_u(s)$ for each $u \in \Sigma^n$. We define

$$\phi_u(s) = \sum_{\mathbf{i}:u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})}, \tag{2}$$

¿Qué enfoque distinto plantea el nuevo kernel?

¿Qué enfoque distinto plantea el nuevo kernel?

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

¿Qué enfoque distinto plantea el nuevo kernel?

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

$$\begin{aligned} K_n(s,t) &= \sum_{u\in\Sigma^n} \langle \phi_u(s)\cdot\phi_u(t)\rangle = \sum_{u\in\Sigma^n} \sum_{\mathbf{i}:u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})} \sum_{\mathbf{j}:u=t[\mathbf{j}]} \lambda^{l(\mathbf{j})} \\ &= \sum_{u\in\Sigma^n} \sum_{\mathbf{i}:u=s[\mathbf{i}]} \sum_{\mathbf{j}:u=t[\mathbf{j}]} \lambda^{l(\mathbf{i})+l(\mathbf{j})}. \end{aligned}$$

Cálculo recursivo con función auxiliar

Cálculo recursivo con función auxiliar

$$
\begin{aligned}
K_i'(s,t) &= \sum_{u \in \Sigma^i} \sum_{\mathbf{i}:u=s[\mathbf{i}]} \sum_{\mathbf{j}:u=t[\mathbf{j}]} \lambda^{|s|+|t|-i_1-j_1+2}, \\
i &= 1, \ldots, n-1,
\end{aligned}
$$

Cálculo recursivo con función auxiliar

$$
\begin{aligned}
K'_i(s,t) &= \sum_{u\in\Sigma^i} \sum_{\mathbf{i}:u=s[\mathbf{i}]} \sum_{\mathbf{j}:u=t[\mathbf{j}]} \lambda^{|s|+|t|-i_1-j_1+2}, \\
i &= 1,\ldots,n-1,
\end{aligned}
$$

$$
\begin{aligned}
K'_0(s,t) &= 1, \textit{ for all } s,t, \\
K'_i(s,t) &= 0, \textit{ if } \min(|s|,|t|) < i, \\
K_i(s,t) &= 0, \textit{ if } \min(|s|,|t|) < i, \\
K'_i(sx,t) &= \lambda K'_i(s,t) + \sum_{j:t_j=x} K'_{i-1}(s,t[1:j-1])\lambda^{|t|-j+2}, \\
& i = 1,\ldots,n-1, \\
K_n(sx,t) &= K_n(s,t) + \sum_{j:t_j=x} K'_{n-1}(s,t[1:j-1])\lambda^2.
\end{aligned}
$$

Cálculo más eficiente

Cálculo más eficiente

$$K''_i(sx,t) = \sum_{j:t_j=x} K'_{i-1}(s,t[1:j-1])\lambda^{|t|-j+2}$$

and observing that we can then evaluate $K'_i(s,t)$ with the $O(|s||t|)$ recursion,

$$K'_i(sx,t) = \lambda K'_i(s,t) + K''_i(sx,t).$$

Now observe that $K''_i(sx,tu) = \lambda^{|u|}K''_i(sx,t)$, provided x does not occur in u, while

$$K''_i(sx,tx) = \lambda\left(K''_i(sx,t) + \lambda K'_{i-1}(s,t)\right).$$

These observations together give an $O(|s||t|)$ recursion for computing $K''_i(s,t)$. Hence, we can evaluate the overall kernel in $O(n|s||t|)$ time.

Cálculo aún más eficiente

Cálculo aún más eficiente

- La idea es implementar una técnica al estilo DP.

Cálculo aún más eficiente

- La idea es implementar una técnica al estilo DP.
- Se toma un enfoque bottom-up: el sistema emplea una estrategia de fragmentación donde comienza con un pequeño subconjunto de datos y gradualmente aumenta el tamaño del conjunto de entrenamiento. La clave del éxito es asegurar que solo se incluyan en el siguiente fragmento aquellos puntos que no cumplen con un criterio de margen en la hipótesis actual. Es decir, se calcula solo lo necesario.

Cálculo aún más eficiente

- La idea es implementar una técnica al estilo DP.
- Se toma un enfoque bottom-up: el sistema emplea una estrategia de fragmentación donde comienza con un pequeño subconjunto de datos y gradualmente aumenta el tamaño del conjunto de entrenamiento. La clave del éxito es asegurar que solo se incluyan en el siguiente fragmento aquellos puntos que no cumplen con un criterio de margen en la hipótesis actual. Es decir, se calcula solo lo necesario.
- Y claramente las entradas del kernel calculadas se guardan para su reutilización en diferentes divisiones de datos.

Cálculo aún más eficiente

- La idea es implementar una técnica al estilo DP.
- Se toma un enfoque bottom-up: el sistema emplea una estrategia de fragmentación donde comienza con un pequeño subconjunto de datos y gradualmente aumenta el tamaño del conjunto de entrenamiento. La clave del éxito es asegurar que solo se incluyan en el siguiente fragmento aquellos puntos que no cumplen con un criterio de margen en la hipótesis actual. Es decir, se calcula solo lo necesario.
- Y claramente las entradas del kernel calculadas se guardan para su reutilización en diferentes divisiones de datos.
- Matrices N-gram.

Aproximación de matrcies N-gram

Aproximación de matrcies N-gram

The approximation approach we adopt is based on a more general empirical kernel map introduced by Schölkopf et al. (1999). We consider the special case when the set of vectors is chosen to be orthogonal. Assume we have some training points $(x_i, y_i) \in X \times Y$, and some kernel function $\mathcal{K}(x, z)$ corresponding to a feature space mapping $\phi : X \mapsto \mathcal{F}$ such that $\mathcal{K}(x, z) = \langle \phi(x), \phi(z) \rangle$. Consider a set S of vectors $S = \{s_i \in \mathcal{X}\}$. If the cardinality of S is equal to the dimensionality of the space $\mathcal{F}$ and the vectors $\phi(s_i)$ are orthogonal (i.e. $\mathcal{K}(s_i, s_j) = C\delta_{ij})^1$, then the following is true:

$$\mathcal{K}(x, z) = \frac{1}{C} \sum_{s_i \in S} \mathcal{K}(x, s_i)\mathcal{K}(z, s_i). \tag{3}$$

Aproximación de matrices N-gram

Aproximación de matrices N-gram

This follows from the fact that $\phi(x) = \frac{1}{C}\sum_{s_i \in S} \mathcal{K}(x, s_i)\phi(s_i)$ and $\phi(z) = \frac{1}{C}\sum_{s_j \in S} \mathcal{K}(z, s_j)\phi(s_j)$, so that

$$\begin{aligned} \mathcal{K}(x, z) &= \frac{1}{C^2} \sum_{s_i, s_j \in S} \mathcal{K}(x, s_i)\mathcal{K}(z, s_j) C\delta_{ij} \\ &= \frac{1}{C} \sum_{s_i \in S} \mathcal{K}(x, s_i)\mathcal{K}(z, s_i). \end{aligned}$$

If instead of forming a complete orthonormal basis, the cardinality of $\tilde{S} \subseteq S$ is less than the dimensionality of X or the vectors s_i are not fully orthogonal, then we can construct an approximation to the kernel $\mathcal{K}$:

$$\mathcal{K}(x, z) \approx \sum_{s_i \in \tilde{S}} \mathcal{K}(x, s_i)\mathcal{K}(z, s_i). \tag{4}$$

Aproximación de matrices N-gram - cálculo de subset

Aproximación de matrices N-gram - cálculo de subset

- Se elije un tamaño de subcadena n.

Aproximación de matrices N-gram - cálculo de subset

- Se elije un tamaño de subcadena n.
- Se enumeran todas las posibles cadenas contiguas de longitud n.

Aproximación de matrices N-gram - cálculo de subset

- Se elije un tamaño de subcadena n.
- Se enumeran todas las posibles cadenas contiguas de longitud n.
- Se seleccionan las x cadenas de longitud n que ocurren con más frecuencia en el conjunto de datos, y esto forma el subconjunto.

Resultados I

Resultados I

Category	Kernel	λ	F1		Precision		Recall	
			Mean	SD	Mean	SD	Mean	SD
earn	NGK	0	0.944	0.026	0.992	0.013	0.903	0.051
	SSK	0.01	**0.946**	0.028	0.992	0.013	0.905	0.052
		0.03	0.946	0.028	0.992	0.013	0.905	0.052
		0.05	0.944	0.026	0.992	0.013	0.903	0.051
		0.07	0.944	0.026	0.992	0.013	0.903	0.051
		0.09	0.944	0.026	0.992	0.013	0.902	0.051
		0.1	0.944	0.026	0.992	0.013	0.903	0.051
		0.3	0.943	0.029	0.992	0.013	0.900	0.055
		0.5	0.936	0.0130	0.992	0.014	0.888	0.067
		0.7	0.928	0.040	0.994	0.012	0.873	0.062
		.9	0.914	0.050	0.989	0.020	0.853	0.075
	W-K		0.925	0.014	0.989	0.014	0.867	0.057
acq	NGK	0	**0.882**	0.038	0.912	0.041	0.856	0.051
	SSK	0.01	0.873	0.040	0.910	0.040	0.840	0.050
		0.03	0.878	0.040	0.908	0.040	0.852	0.057
		0.05	**0.882**	0.037	0.912	0.040	0.856	0.054
		0.07	0.873	0.044	0.910	0.041	0.840	0.063
		0.09	0.863	0.043	0.908	0.041	0.824	0.063
		0.1	0.871	0.043	0.903	0.038	0.844	0.069
		0.3	0.870	0.040	0.911	0.051	0.836	0.061
		0.5	0.867	0.038	0.914	0.042	0.828	0.067
		0.7	0.805	0.050	0.935	0.046	0.712	0.078
		0.9	0.735	0.073	0.850	0.064	0.652	0.092
	WK		0.802	0.033	0.843	0.067	0.768	0.057

Table 3: The performance (F1, precision, recall) of SVM with SSK, NGK and WK for Reuters categories earn and acq. Results illustrate the impact of varying λ on performance of SSK. The results are averaged over 10 runs of the techniques. We also report standard deviation.

Resultados II

Resultados II

Category	Kernel	Length	F1 Mean	F1 SD	Precision Mean	Precision SD	Recall Mean	Recall SD
earn	SSK	3	0.925	0.036	0.981	0.030	0.878	0.057
		4	0.932	0.029	0.992	0.013	0.888	0.052
		5	0.936	0.036	0.992	0.013	0.888	0.067
		6	0.936	0.033	0.992	0.013	0.888	0.060
		7	0.940	0.035	0.992	0.013	0.900	0.064
		8	0.934	0.033	0.992	0.010	0.885	0.058
		10	0.927	0.032	0.997	0.009	0.868	0.054
		12	0.931	0.036	0.981	0.025	0.888	0.058
		14	0.936	0.027	0.959	0.033	0.915	0.041
	NGK	3	0.919	0.035	0.974	0.036	0.873	0.062
		4	0.943	0.030	0.992	0.013	0.900	0.055
		5	**0.944**	0.026	0.992	0.013	0.903	0.051
		6	0.943	0.030	0.992	0.013	0.900	0.055
		7	0.940	0.035	0.992	0.013	0.895	0.064
		8	0.940	0.045	0.992	0.013	0.895	0.063
		10	0.932	0.032	0.990	0.015	0.885	0.053
		12	0.917	0.033	0.975	0.024	0.868	0.053
		14	0.923	0.034	0.973	0.033	0.880	0.055
	WK		0.925	0.033	0.989	0.014	0.867	0.057
acq	SSK	3	0.785	0.040	0.863	0.060	0.724	0.064
		4	0.822	0.047	0.898	0.045	0.760	0.068
		5	0.867	0.038	0.914	0.042	0.828	0.057
		6	0.876	0.051	0.934	0.043	0.828	0.080
		7	0.864	0.045	0.920	0.046	0.816	0.063
		8	0.852	0.049	0.918	0.051	0.796	0.064
		9	0.820	0.056	0.903	0.053	0.756	0.089
		10	0.791	0.067	0.848	0.072	0.744	0.083
		12	0.791	0.067	0.848	0.072	0.744	0.083
		14	0.774	0.042	0.819	0.067	0.736	0.043
	NGK	3	0.791	0.043	0.842	0.061	0.748	0.053
		4	0.873	0.031	0.896	0.037	0.852	0.038
		5	**0.882**	0.038	0.912	0.041	0.856	0.051
		6	0.880	0.045	0.923	0.041	0.844	0.072
		7	0.870	0.050	0.904	0.047	0.844	0.085
		8	0.857	0.044	0.897	0.039	0.824	0.071
		10	0.830	0.045	0.887	0.063	0.784	0.071
		12	0.806	0.066	0.850	0.062	0.768	0.079
		14	0.776	0.060	0.814	0.061	0.744	0.076
	W-K		0.802	0.072	0.843	0.067	0.768	0.090

Table 1: The performance (F1, precision, recall) of SVM with SSK, NGK and WK for Reuters categories earn and acq. Results illustrate the effect of the variability of subsequence length on performance. The results are averaged over 10 runs of the techniques. We also report standard deviation.

Resultados III

Resultados III

Category	w_{ng}	w_{sk}	F1		Precision		Recall	
			Mean	SD	Mean	SD	Mean	SD
earn	1	0	**0.944**	0.026	0.992	0.013	0.903	0.051
	0	1	0.936	0.013	0.992	0.014	0.888	0.067
	0.5	0.5	0.944	0.026	0.992	0.013	0.903	0.051
	0.6	0.4	0.941	0.029	0.992	0.013	0.898	0.055
	0.7	0.3	0.941	0.029	0.992	0.013	0.898	0.055
	0.8	0.2	**0.944**	0.030	0.992	0.013	0.903	0.060
	0.9	0.1	0.943	0.0260	0.992	0.013	0.900	0.050
acq	1	0	**0.882**	0.038	0.912	0.041	0.856	0.051
	0	1	0.867	0.038	0.914	0.042	0.828	0.067
	0.5	0.5	0.865	0.035	0.917	0.045	0.820	0.051
	0.6	0.4	0.878	0.031	0.908	0.036	0.852	0.046
	0.7	0.3	0.868	0.047	0.909	0.040	0.832	0.073
	0.8	0.2	0.875	0.033	0.913	0.050	0.844	0.058
	0.9	0.1	0.875	0.040	0.910	0.040	0.844	0.058
crude	1	1	0.937	0.048	0.968	0.045	0.913	0.083
	0	1	0.936	0.045	0.979	0.033	0.900	0.078
	0.5	0.5	**0.940**	0.0360	0.987	0.027	0.900	0.072
	0.6	0.4	0.937	0.040	0.980	0.032	0.900	0.072
	0.7	0.3	0.934	0.057	0.994	0.020	0.887	0.105
	0.8	0.2	0.928	0.057	0.981	0.042	0.887	0.104
	0.9	0.1	0.926	0.063	1.000	0.000	0.867	0.104
corn	1	0	0.847	0.103	0.912	0.092	0.800	0.141
	0	1	0.779	0.104	0.886	0.094	0.700	0.125
	0.5	0.5	0.843	0.095	0.929	0.085	0.750	0.127
	0.6	0.4	0.836	0.091	0.943	0.085	0.760	0.126
	0.7	0.3	0.827	0.098	0.916	0.083	0.760	0.126
	0.8	0.2	0.831	0.093	0.930	0.084	0.760	0.126
	0.9	0.1	**0.849**	0.092	0.943	0.084	0.780	0.123

Table 6: The performance (F1, precision, recall) of SVM with combined kernels for Reuters categories earn, acq, corn and crude. The SSK and NGK are combined. The results are averaged and standard deviation is also given.

Resultados IV

Resultados IV

Category	WK	NGK			Approximated SSK		
		n			k		
		3	4	5	3	4	5
earn	0.982	0.982	0.984	0.974	0.970	0.970	0.970
acq	0.948	0.919	0.932	0.887	0.850	0.880	0.880
money-fx	0.775	0.744	0.757	0.692	0.700	0.760	0.740
grain	0.930	0.852	0.840	0.758	0.800	0.820	0.800
crude	0.880	0.857	0.848	0.640	0.820	0.840	0.840
trade	0.761	0.774	0.779	0.767	0.700	0.730	0.730
interest	0.691	0.708	0.719	0.503	0.630	0.660	0.690
ship	0.797	0.657	0.626	0.321	0.610	0.650	0.530
wheat	0.870	0.803	0.797	0.629	0.780	0.790	0.820
corn	0.895	0.761	0.610	0.459	0.630	0.630	0.680

Table 9: F1 numbers for SVM with WK, NGK and SSK for top-ten Reuters categories.

Índice

1. ¿Por qué es importante la tarea de la clasificación de textos?
2. Un pequeño repaso sobre kernels
    - El "truquito" del kernel
    - Kernels tradicionales vs kernels sobre conjuntos generales
3. Sobre el Paper
    - ¿Qué había hasta el momento?
    - ¿Qué enfoque distinto plantea el nuevo kernel?
    - ¿Cómo se hace eficiente el cálculo?
    - Resultados
4. Conclusiones

Conclusiones

Conclusiones

- Vemos muy buenos resultados, en principal en comparación a la técnica bag of words, pero para datasets pequeños.

Conclusiones

- Vemos muy buenos resultados, en principal en comparación a la técnica bag of words, pero para datasets pequeños.
- Para datasets grandes los resultados no fueron tan alentadores.

Conclusiones

- Vemos muy buenos resultados, en principal en comparación a la técnica bag of words, pero para datasets pequeños.
- Para datasets grandes los resultados no fueron tan alentadores.
- Se especula que los buenos resultados en los datasets pequeños se debe a que el kernel está funcionando como stemming, siendo está la razón por la cual no escala bien a datasets más grandes.