IIC2613 Inteligencia Articial - Tarea 3

Rut	UC	Nombre
18501880-6	13634941	Nicolás Gebauer Martínez

Información general

Primero se debe leer el readme

Experimentos con Descenso de Gradiente

Primero se executó el programa `DescGrdt1` con los valores por defecto `old_min = 10` y `learningRate = 0.1`. Luego se modificó la tasa de aprendizaje a valores menores, en particula se probó con `0.08, 0.05, 0.04, 0.01, 0.005, 0.001`.
Se observa que al disminuir la tasa de aprendizaje la velocidad con la que se desciende hacia el mínimo de la función es cada vez menor. En particular, con una tasa de aprendizaje de `0.001` se demora minutos, mientras que con `0.1` se hace en unos pocos segundos. También es de destacar que la disminución de la tasa de aprendizaje se hace mucho más evidente al estar evaluando valores cercanos al mínimo (`x<1`).
Luego se aumento el valor de la tasa de aprendizaje. Se probaron los valores `0.2, 0.4, 0.6, 0.8, 1`. Se observa que la velocidad de descenso es mayor al aumentar la tasa de aprendizaje. Se destaca que para los valores `0.6` y `0.8` se observa que el descenso se "pasa de largo". Dado que se aumentó tanto la tasa de aprendizaje se logró dar un salto que pasó de `x=10` a un `x~-5`. Luego volvió a converger hacia `x=0`.

De aqui se puede concluir ventajas y desventajas de el valor de la tasa de aprendizaje. Con un mayor valor se tiende más rápido al mínimo pero se pueden dar saltos muy grandes, pasandose del mínimo. En cambio, si la tasa de aprendizaje es pequeña se evita pasar de largo pero la convergencia es mucho más lenta.

Luego se executó el programa `DescGrdt2` con los valores por defecto `old_min = 1` y `learningRate = 0.1`. Se observa que efectivamente se tiende al mínimo local Se executó con distintas combinaciones de punto inicial y tasas de aprendizaje, de las cuales se destacan las siguientes:

`old_min`	`learningRate`	**mínimo**
1	0.1	-2.09
2.2	0.1	4.19
1	4	-6938.73
2.2	3.6	-34119.7
2.2	4	-820265.65

Se puede observar que el punto inicial y la tasa de aprendizaje tienen un efecto muy importante en el resultado final, no solo en que tan rápido se calcula. Si se cambia el punto inicial se termina obteniendo un mínimo local. Además, si la tasa de aprendizaje es muy grande se puede terminar en un mínimo local al que se quería en un principio.

Análisis de Sentimientos en Críticas de Películas

Word2Vec

Se ejecutaron las comparaciones pedidas, más las extras `car + water - wheels` con la intención de obtener `boat` y ver la similitud de `man` con `men`. Los resultados son los siguientes:

Comparing most similar to positives ['woman', 'king'] and negatives ['man']
(u'queen', 0.7118192315101624)
(u'monarch', 0.6189674139022827)
(u'princess', 0.5902431011199951)
(u'crown_prince', 0.5499460697174072)
(u'prince', 0.5377321839332581)
(u'kings', 0.5236844420433044)
(u'Queen_Consort', 0.5235946178436279)
(u'queens', 0.5181134343147278)
(u'sultan', 0.5098593235015869)
(u'monarchy', 0.5087412595748901)

Comparing most similar to positives ['paris', 'italy'] and negatives ['france']
(u'lohan', 0.5069675445556641)
(u'madrid', 0.4818429350852966 3)
(u'heidi', 0.4799900949001312)
(u'real_madrid', 0.4753322899341583)
(u'florence', 0.4682057499885559)
(u'diego', 0.467273086309433)
(u'ronnie', 0.4672326445579529)
(u'juventus', 0.4672061502933502)
(u'joel', 0.4653723835945129 4)
(u'huntelaar', 0.4635899609947205)

Comparing most similar to positives ['car', 'water'] and negatives ['wheels']
(u'sewage', 0.5073274374008179)
(u'groundwater', 0.4882017374038696 3)
(u'Floridan_aquifer', 0.4697296917438507)
(u'aquifers', 0.4586605429649353)

> (u'Water', 0.4582120180130005)
> (u'underground_aquifer', 0.4578779935836792)
> (u'tapwater', 0.453736811876297)
> (u'freshwater', 0.4533122479915619)
> (u'shallow_aquifer', 0.4453983306884765 6)
> (u'Elephant_Butte_Reservoir', 0.4452925026416778 6)
>
> Comparing similarity of breakfast with cereal
> 0.365095440625
>
> Comparing similarity of class with course
> 0.22938951003
>
> Comparing similarity of man with men
> 0.548976303226

Se puede observar que efectivamente se cumplen algunas de las relaciones mencionadas en el informe. `rey-hombre+mujer` se parece a `reina` con un `71%` de similitud, lo cual es un número muy elevado. Otra similitud alta fue la de `men` con `man`. Llama la atención que estos resultados son distintos. Se tenía la intención de que auto - ruedas + agua diera bote, pero los resultados no fueron favorables.

PolarityExamples.txt y MovieReviewExamples.txt

Al observar las reseñas de películas se destaca que estas tienden a ser un conjunto de comentarios cortos, a diferencia de por ejemplo frases más largas y estructuradas. Sin embargo, siguen siendo comentarios con un orden más formal que los del archivo *polarity*. Este otro tiene mensajes informales y desordenados que incluyen, por ejemplo, comentarios sarcásticos que podrían ser más difíciles de entender para una inteligencia artificial, dado que el sentido del mensaje es el contrario del significado que se extrae de las palabras usadas.

SVM

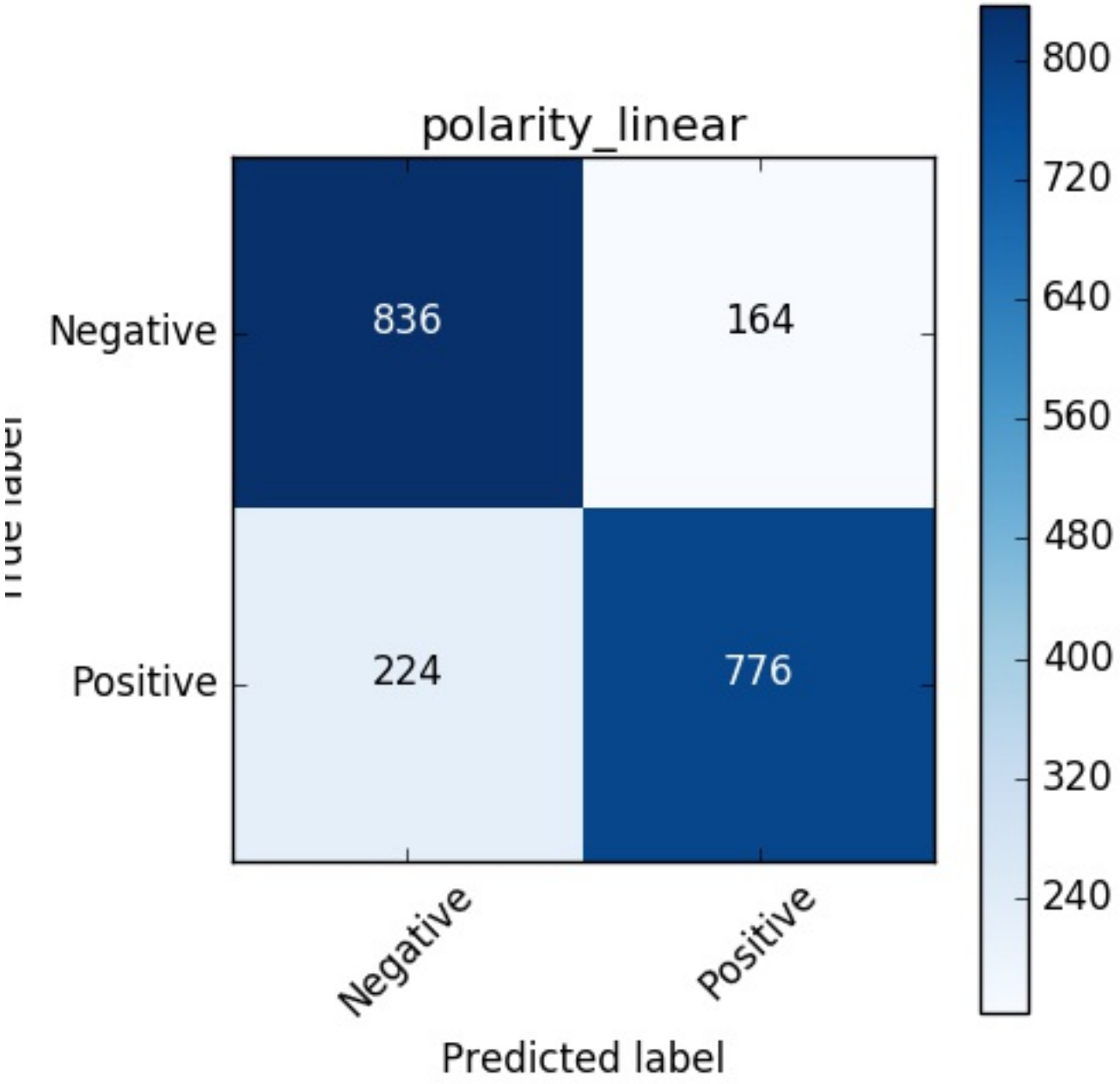
Se probo el rendimiento de clasificación obtenido por un SVM sobre la codificación de frases descrita. Los resultados obtenidos se guardaron en el archivo 3.4.5.log y se presentan ordenados a continuación

prueba	***accuracy***	***precision***
polarity_linear	0.806	0.807105580088318
polarity_rbf	0.7105	0.7277214334009465
movies_linear	0.83712	0.8371272582150184
movies_rbf	0.747	0.7488890415426257

Se puede observar en ambos casos que el kernel lineal tuvo un mejor resultado que el rbf. Con precisiones sobre 80% se puede decir que los resultados fueron bastante buenos.

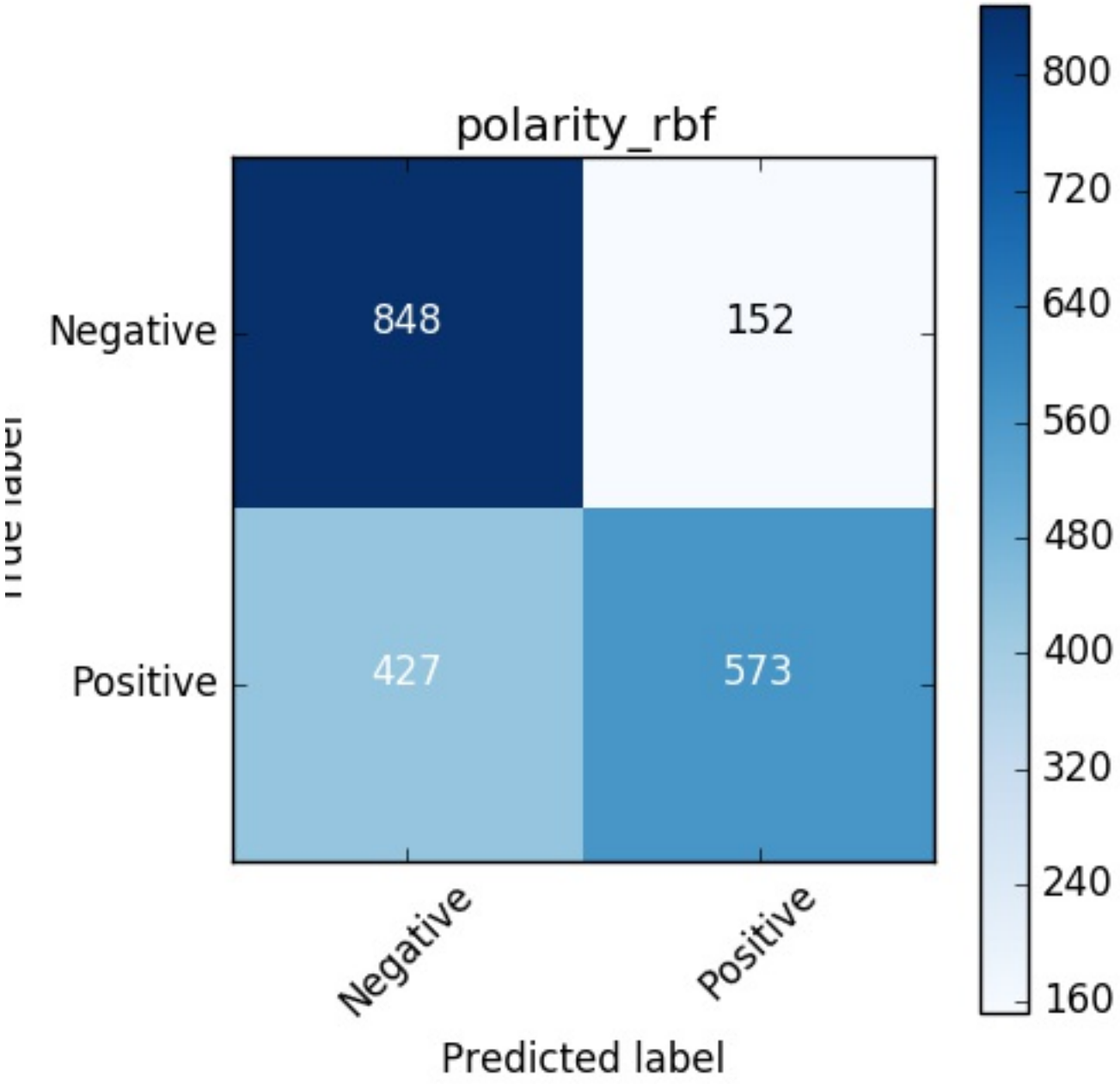
Matriz confusión

En lo que a las matrices de confusión respecta se obtenieron las siguientes:



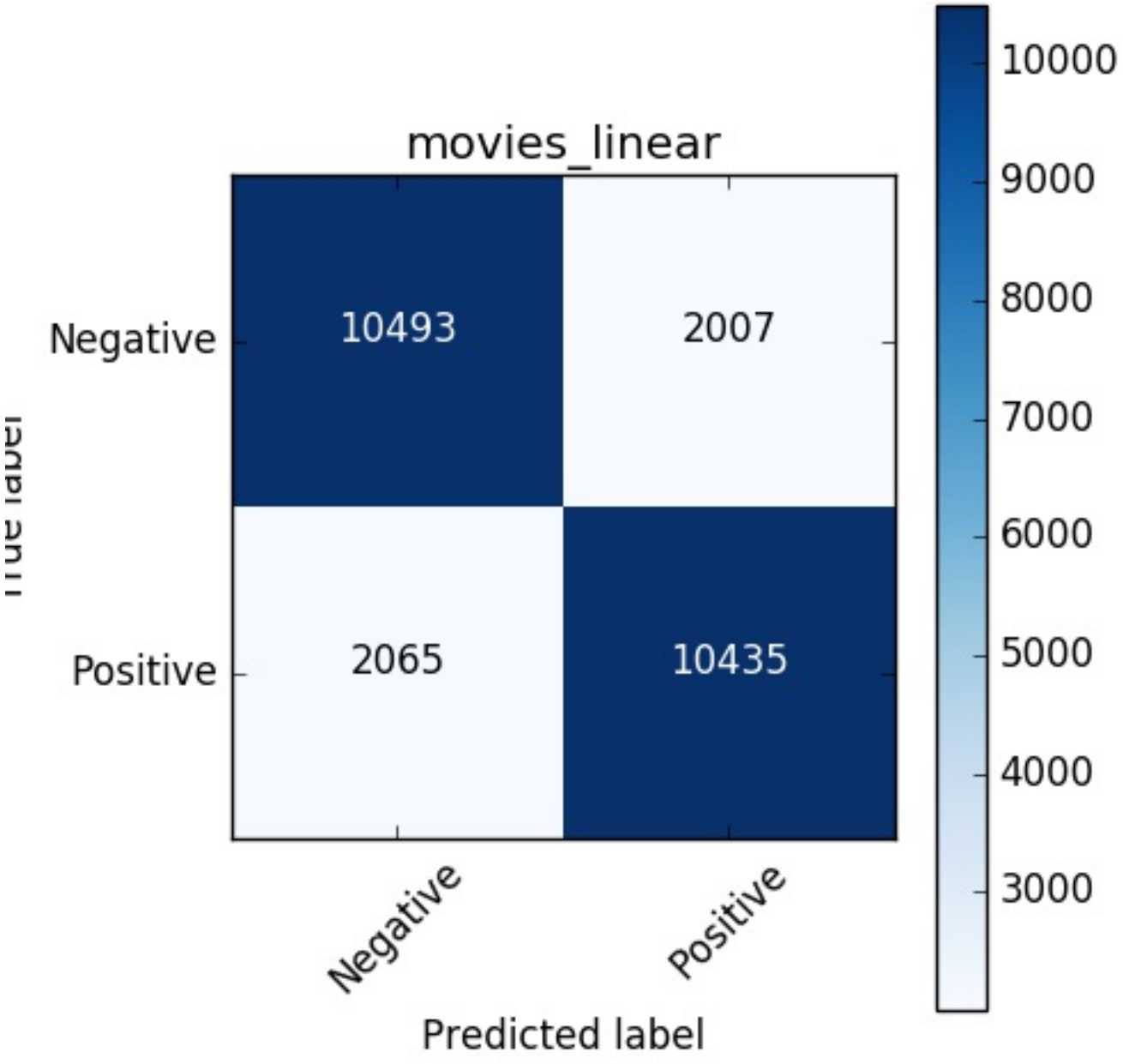
Rendimiento SVM con los datos de polaridad con kernel lineal

Se observa que la gran mayoría de errores ocurren por positivos que fueron clasificados como negativos, el doble de lo que ocurrió con los negativos, un ~33% más.



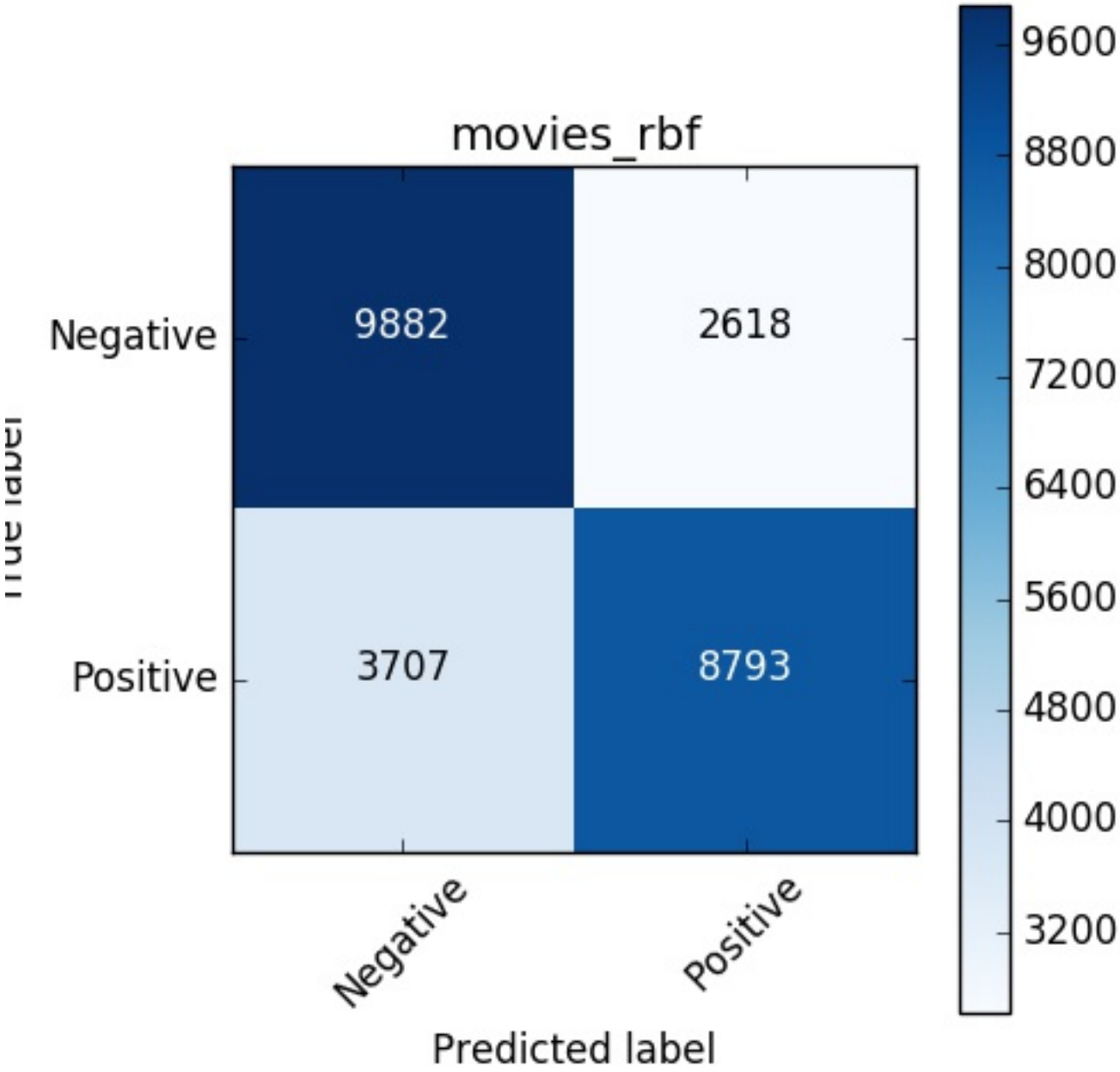
Rendimiento SVM con los datos de polaridad con kernel rbf

Se observa que la tendencia con el modelo linear se repite, la mayoría de errores son falsos negativos. Se destaca que esta tendencia se incrementa, teniendo un ~42% de error en la predicción de positivos, lo cual es un error preocupante.



Rendimiento SVM con los datos de perlícula con kernel linear

Se destaca que la tendencía de falsos negativos no se mantiene para este modelo. Su rendimiento es alto (~83%) y no hay una diferencia notable entre los errores positivos y negativos, a diferencia de las matrices anteriores.



Rendimiento SVM con los datos de perlícula con kernel rbf

Se observa que hay una diferencia de 100 entre los falsos positivos y los falsos negativos. La tendencia observada con las polaridades se vuelve a cumplir en este caso. Esto puede deberse a que es común que criticas positivas incluyan comentarios negativos. Por ejemplo, hay críticas donde se critican los aspectos negativos de una película y se termina diciendo que a pesar de ellos la película es muy buena, lo cual puede ser causante de estos errores.

Mejoras

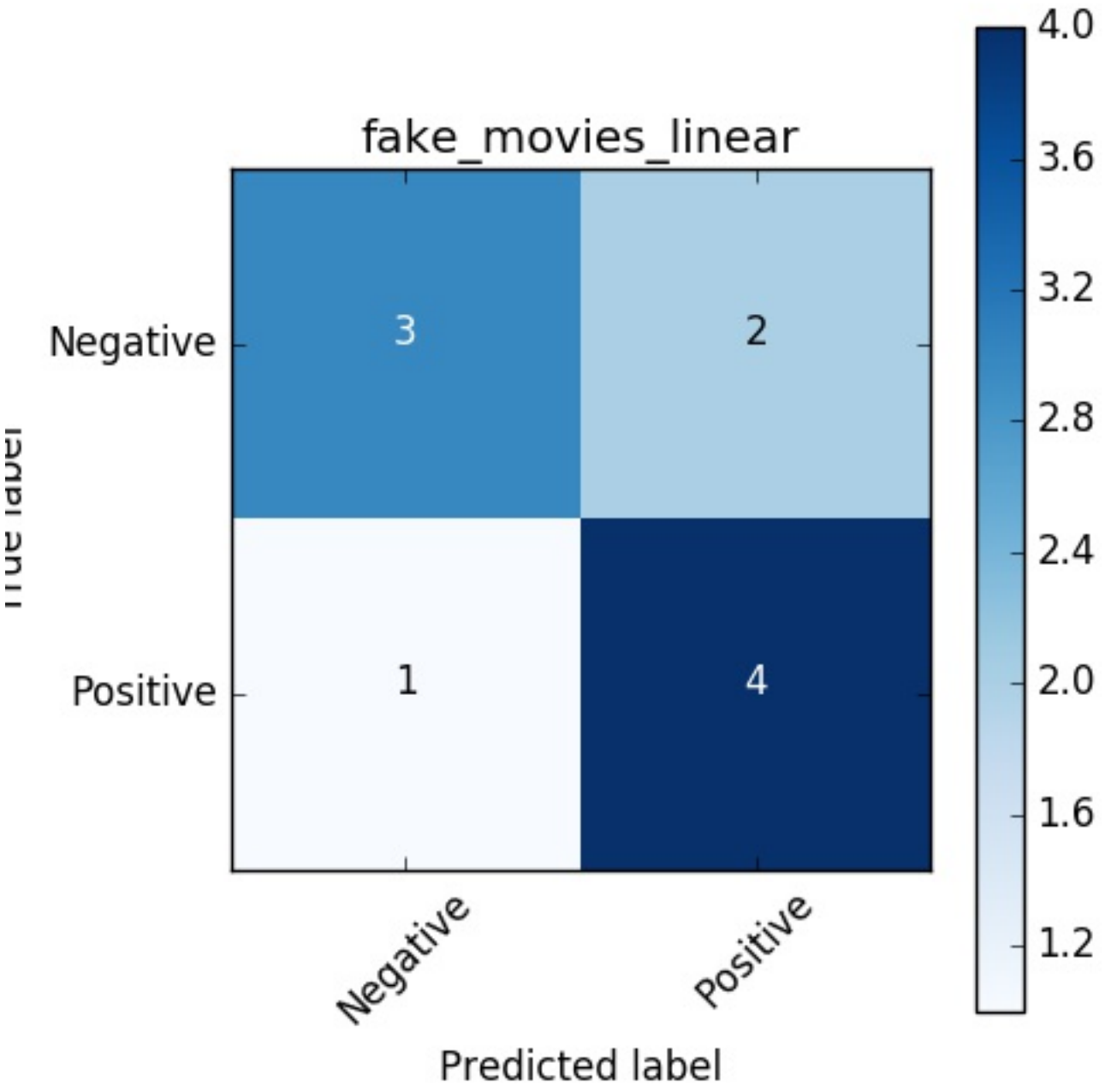
Lo primero que se podría considerar es variar el coeficiente de penalización `C`, el cual se mantuvo fijo en `1`. Como se vio en la tarea anterior, al variar dicho coeficiente los resultados cambian, por lo que se podrían mejorar encontrando un mejor `C`.
Otro tema relevante es como se codifican las palabras con `word2vec`.
En este caso se tiene un sistema de codificación que fue entrenado con noticias. Por lo tanto, el sistema esta "más ajustado" para el uso de las palabras y frases en el contexto de noticias. Luego se utiliza para evaluar comentarios de películas, un contexto en el cual las palabras se usan distinto que en las noticias. Es decir, si se tuviera un modelo `word2vec` entrenado con críticas de películas la predicción debería ser mejor (una forma de 'sobre ajuste').

Críticas inventadas

Se volvió a probar el rendimiento de clasificación con SVM con las críticas inventadas, las cuales se encuentran en fake_reviews. Se probó con ambos *kernels* ya que se ejecutaron las pruebas al mismo tiempo. El resultado pedido corresponde al obtenido con el *kernel* lineal (el que tuvo mejor rendimiento antes). Los resultados obtenidos se guardaron en el archivo 3.4.6.log y se presentan ordenados a continuación:

prueba	***accuracy***	***precision***
movies_linear	0.7	0.7083333333333333
movies_rbf	0.6	0.7777777777777778

Se observa que nuevamente el *kernel* linear obtiene mejores resultados que el *kernel* rdf. Las matrices de confusión obtenidas son las siguientes:



Rendimiento SVM con las críticas inventadas con kernel linear

Se observa que hubo dos falsos positivos y un falso negativo. Se destaca que entre los comentarios inventados hay dos irónicos, que comienzan con palabras que se usan en comentarios positivos para finalizar con un comentario negativo.
Ejemplo:

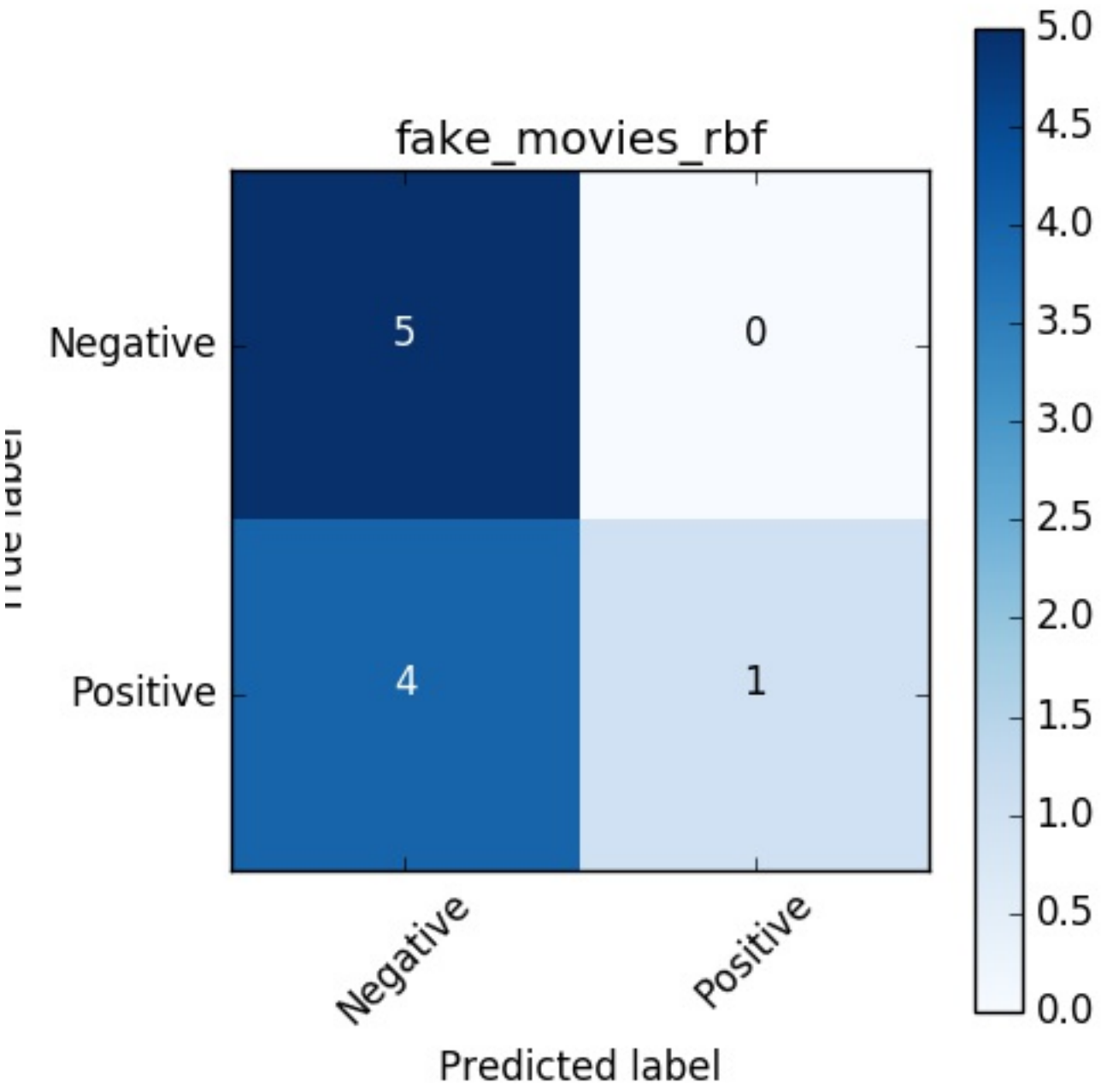
> They said they had the best actors in the world.I didn't see one

Se podría decir que críticas como esta tienen una mayor tendencia al

error. Sobre el falso negativo, se tienen dos comentarios que podrían tener un contenido negativo:

> Altought it was slow, the movie was pretty good. The best I've seen It made me cry. I haven't cried in years. It was so beautifull

Por lo que se puede concluir que el mezclar palabras positivas con negativas aumenta el error, pero aún así logra identificar correctamente algunos casos.



Rendimiento SVM con las críticas inventadas con kernel rbf

Se observa que se mantiene la tendencia de falsos negativos. Aquí llama más la atención ya que en los comentarios positivos hay solo 2 que se podrían interpretar como negativos. El algoritmo interpeto negativamente alguno de los siguiente comentarios:

> I just can't believe how much this movie reflects my own life. Made me rethink everything I've done. The actors played their roles perfectly. 10/10
> It was just glorious
> I don't think I'll ever see a movie as good as this one

Los cuales son positivos. RBF fue muy pobre en esta situación.