UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

CLASIFICACIÓN DE POLARIDAD MONO Y TRANSDOMINIO UTILIZANDO STRING KERNELS

MÁSTER UNIVERSITARIO EN INTELIGENCIA ARTIFICIAL,
RECONOCIMIENTO DE FORMAS E IMAGEN DIGITAL

Autora
Rosa Mª Giménez Pérez

Directores
Paolo Rosso
Marc Franco Salvador

13 de septiembre de 2018

Abstract

The polarity classification task, also known as sentiment classification task, has as objective to automatically deciding whether a subjective text is positive or negative. The classic approach, called single-domain polarity classification, contemplates the classification of texts in the same domain to which the texts used in the training phase belong to. In contrast, in the cross-domain polarity classification approach, the texts to be classified belong to a different domain from those used in the training phase.

The study of this last approach has special interest, since it allows the classification of texts on domains in which there are no labeled samples. In general, this has been tackled using classical domain adaptation methods. These methods employ texts from both the source and target domains to detect pivot features.

In this thesis the state of the art in the single and cross-domain polarity classification tasks is described. We also propose a method that does not require the aforementioned domain adaptation techniques and, therefore, texts from the target domain in the training phase are not necessary. This method, which is based on the use of String Kernels functions, detects the lexical characteristics that define the polarity in a text and map them onto a domain independent space by means of Kernel Discriminant Analysis algorithm. Throughout this work, its behavior is studied to solve the problem of single and cross-domain polarity classification in different languages and alphabets. Experimental results obtained are very promising and show the strong potential of the String Kernels independently from the language.

Resumen

El problema de la clasificación de la polaridad, también conocido como clasificación del sentimiento, tiene como objetivo principal conseguir identificar de manera automática cuando un texto subjetivo es positivo o negativo. El enfoque clásico, denominado clasificación de la polaridad monodominio, contempla la clasificación de textos en el mismo dominio al cual pertenecen los textos utilizados en la fase de entrenamiento. Por contra, en el enfoque transdominio de la clasificación de la polaridad, los textos a clasificar pertenecen a un dominio diferente a los empleados en la fase de entrenamiento.

El estudio de este último enfoque es de especial interés, puesto que permite la clasificación de textos sobre dominios en los que no existen muestras etiquetadas. Comúnmente, se ha tratado de dar solución a este enfoque con técnicas que requerían de métodos de adaptación del dominio. Estos métodos utilizan textos tanto de los dominios fuente como objetivo para detectar características pivote.

En esta tesis se describe el estado del arte en la clasificación de la polaridad monodominio y transdominio. También se propone un método que no precisa de las técnicas de adaptación del dominio mencionadas y, por tanto, no son necesarios textos del dominio objetivo en la fase de entrenamiento. Este método, basado en el uso de funciones String Kernels, detecta las peculiaridades léxicas que caracterizan la polaridad de un texto y las mapea en un espacio independiente del dominio por medio del algoritmo Kernel Discriminant Analysis. A lo largo del trabajo, se estudia su comportamiento para la resolución del problema de clasificación de la polaridad monodominio y transdominio en diferentes idiomas y alfabetos. Los resultados obtenidos son muy prometedores y muestran el fuerte potencial de los String Kernels independientemente del lenguaje.

Agradecimientos

Me gustaría dedicar un pequeño apartado para agradecer a todas las personas que de algún modo u otro han aportado su grano de arena para ayudarme a finalizar este trabajo final de máster.

En primer lugar, me gustaría agradecer especialmente al profesor Paolo Rosso por instruirme en el campo de la lingüística computacional, por brindarme la oportunidad de realizar este proyecto, por toda su supervisión y ayuda y por su comprensión y escucha ante los problemas que me han surgido durante la realización de este trabajo. De igual manera, quiero agradecer enormemente al doctor Marc Franco Salvador toda su ayuda y tutelación, por haber estado disponible en todo momento para resolver todas mis dudas y por su guía y correcciones sin las cuales habría sido imposible conseguir publicaciones en congresos de alto nivel. A ambos agradezco todo el tiempo y esfuerzo que han invertido en ayudarme.

Por otro lado, quiero agradecer a todos los familiares y amigos que han sabido y querido soportarme durante esta etapa, proporcionándome su apoyo, escuchándome y estando a mi lado. Por último quiero dedicarle a mi padre las últimas palabras de este apartado, quien sé que habría estado orgulloso de ver todo el esfuerzo que he ejercido por conseguir estos últimos logros.

Este trabajo se ha desarrollado en el marco del proyecto de investigación SomEMBED TIN2015-71147-C2-1-P MINECO y por la Generalitat Valenciana bajo la subvención ALMAMATER (PrometeoII/2014/030).

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En este apartado se va a presentar una breve introducción al problema planteado en esta tesis, explicando cuál es la motivación que nos impulsa a resolverlo y los objetivos que se pretenden alcanzar. También se detalla un resumen de la estructura que seguirá el documento.

1.1. Descripción del problema, motivación y objetivos

El procesamiento del lenguaje natural o NLP (de las siglas de *Natural Language Processing* en inglés) es el campo dentro de la informática, la inteligencia artificial y la lingüística computacional que estudia el diseño de algoritmos capaces de procesar lenguajes humanos de forma automática. Según Manning y Schütze [22], el objetivo principal de esta rama es explicar y extraer las características de conversaciones, escritos y otros medios en los diferentes lenguajes que utilizamos las personas para comunicarnos. Este campo contempla tareas de diversa índole como la traducción automática de textos [7], el reconocimiento del habla [30], la detección de plagio [1], la definición del perfil del autor de un texto [31], etc.

Una de las tareas que ha dado pie a numerosos estudios es la categorización o clasificación de textos [39]. Esta tarea tiene como objetivo asignar una o varias categorías de un conjunto predefinido a un texto [18]. En el comienzo de su estudio, este problema se trató de abordar definiendo un conjunto de reglas que caracterizase el lenguaje de forma manual por expertos. A partir de los años 90, las técnicas de aprendizaje automático pasaron a ser el método principal para la resolución de esta tarea [39]. Estas técnicas normalmente parten de un conjunto limitado de documentos con una etiqueta de clase ya definida y ejecutan sobre ellos un proceso de varias etapas que dará como resultado un modelo capaz de asignar una categoría a un documento nuevo no visto previamente. Este enfoque presenta mayores ventajas frente al de la ingeniería del conocimiento: mayor eficacia, ahorros en términos de mano de obra experta y mejor portabilidad entre dominios. En el artículo de Sebastiani [39] se presenta un resumen sobre diferentes técnicas de aprendizaje automático aplicadas al problema de la categorización de textos.

La categorización de textos es un problema muy amplio que engloba tareas muy dispares, por lo que no existe una única solución común que proporcione el mejor resultado para todas ellas. La tarea más común dentro de este problema sería la clasificación por temática del texto. Por ejemplo, dado un conjunto de noticias, dividirlas según la sección a la cual pertenecerían: deportes, política, ciencia, etc. [1]

En las últimas décadas, internet se ha convertido en la herramienta clave para que usuarios y consumidores expresen su opinión sobre determinados productos y servicios. Esto es debido, en gran parte, a su comodidad, fácil acceso y amplio alcance. Es por este motivo por el que blogs, foros, redes sociales, plataformas de opiniones y las propias webs de los proveedores están repletas de enormes cantidades de datos. De estos datos se puede extraer información muy valiosa, tanto para las empresas como para los consumidores, si se consiguen analizar correctamente. En el marco de este

[1] http://qwone.com/ jason/20Newsgroups/

nuevo contexto, las empresas adquieren un especial interés en identificar las opiniones que los consumidores escriben en las diferentes plataformas sobre sus productos y servicios con el objetivo de mejorar sus campañas de marketing, adaptar sus producciones, etc.

Esta tarea particular de analizar una opinión y decidir si está expresando una valoración positiva o negativa se denomina clasificación del sentimiento o polaridad. La clasificación de la polaridad monodominio [27] (abreviado SD, del inglés *single-domain*) hace referencia a la configuración estándar de la clasificación de textos [39]. Esta configuración implica trabajar sobre textos de un mismo dominio. A diferencia de este enfoque, en la clasificación de la polaridad transdominio [3] (abreviado CD, del inglés *cross-domain*) los textos empleados para el entrenamiento del modelo pertenecen a un dominio distinto al de los textos que van a utilizarse en la fase de clasificación. A pesar de que esta tarea podría resolverse con métodos de clasificación de textos comunes, se ha demostrado que es un problema más complicado. Mientras que en un problema de clasificación por temática puede ser suficiente con identificar determinadas palabras clave, el sentimiento expresado en un documento podría estar representado de una manera mucho más sutil debido al uso de figuras retóricas (por ejemplo, ironía [32]), juegos de palabras, etc. A esto hay que sumarle la dificultad añadida que supone la variante CD del problema, donde el vocabulario variará dependiendo del dominio. Por estos motivos, las técnicas basadas en bolsas de palabras o *bag-of-words* (BoW) pueden no ser suficiente.

Los String Kernels (SK) son funciones que miden la similitud entre dos cadenas de caracteres. Técnicas como ésta, que funcionan a nivel de carácter, son capaces de proporcionar buenos resultados en clasificación de textos [21] y han demostrado su capacidad para capturar las peculiaridades léxicas del texto [28][17].

El objetivo principal de este trabajo es estudiar en profundidad los resultados de aplicar SK al problema de clasificación de la polaridad, tratando de dar respuesta a las siguientes preguntas:

- *¿Cuál es el comportamiento de los SK en el problema de clasificación de la polaridad mono y transdominio?*

  A lo largo de esta tesis se estudiará el comportamiento de los SK en estos dos problemas concretos. Para llevar a cabo esta tarea, se realizará una primera fase de evaluación con el corpus Multi-Domain Sentiment Dataset (v. 2.0). [2] Este corpus está formado por diferentes reviews de Amazon en inglés catalogadas según su sentimiento (positivo o negativo). Hasta donde sabemos, éste será el primer estudio donde se utilicen SK a nivel transdominio, lo cual conduce a la siguiente pregunta.

- *¿Puede este tipo de representación ser usado a nivel transdominio sin necesidad de entrenar con los textos del dominio objetivo?*

  Como modelo de clasificación se utilizará el denominado Kernel Discriminant Analysis (KDA) [23][2]. Este modelo está basado en crear un espacio de transformación no lineal. El objetivo en este punto será aclarar si las peculiaridades léxicas capturadas por este enfoque son suficientes para definir la polaridad del texto independientemente del dominio que se haya usado en la fase de entrenamiento.

- *¿Son las conclusiones obtenidas extrapolables a otros idiomas?*

  Partiendo de los resultados obtenidos, se llevará a cabo una segunda fase de evaluación con el corpus Cross-Lingual Sentiment (CLS). [3] De nuevo, este corpus está formado por reviews de Amazon catalogadas según su sentimiento pero, además del inglés, contempla comentarios también en alemán, francés y japonés. Con el fin de estudiar el comportamiento de los SK en mayor profundidad, el objetivo en esta segunda fase será llevar a cabo los experimentos necesarios para su estudio con estos nuevos lenguajes a fin de descubrir su impacto en la selección de parámetros clave (como la longitud de cadena) según el alfabeto e idioma empleado.

[2] https://www.cs.jhu.edu/ mdredze/datasets/sentiment/

[3] https://www.uni-weimar.de/en/media/chairs/computer-science-department/webis/data/corpus-webis-cls-10/

1.2. Estructura de la tesis

El resto del documento consta de los siguientes capítulos detallados a continuación:

- **Capítulo 2: Estado del arte**

  En este capítulo se describirá el estado del arte para el problema de clasificación de la polaridad de textos aplicado al corpus Multi-Domain Sentiment Dataset ya mencionado. Se describirán los métodos que actualmente proporcionan los mejores resultados tanto en el ámbito mono como transdominio. También se dará una explicación detallada de las herramientas y técnicas empleadas para calcular los baselines para los experimentos realizados con este corpus y con el dataset CLS.

- **Capítulo 3: String Kernels**

  En este apartado se describen en profundidad los diferentes tipos de SK empleados en este trabajo: *spectrum*, *presence* e *intersection*. También se da una explicación más detallada del algoritmo de transformación del espacio KDA empleado.

- **Capítulo 4: Evaluación y resultados**

  Este capítulo muestra y analiza los resultados obtenidos de aplicar las diferentes variantes de SK utilizando como modelo de transformación KDA. En una primera fase de evaluación, se obtienen resultados utilizando el Multi-Domain Sentiment dataset. En la segunda, se analiza el comportamiento de los SK para diferentes idiomas utilizando el Cross-Lingual Sentiment dataset.

- **Capítulo 5: Conclusiones**

  A partir de los resultados evaluados en el capítulo anterior, se plantean las conclusiones extraídas y se proponen nuevas líneas de investigación partiendo de ellas.

- **Apéndice A: Publicaciones**

  Este apéndice presenta las publicaciones derivadas de este trabajo de investigación y sus resultados.

Capítulo 2

Estado del arte

En este capítulo se va a hacer una introducción al estado del arte para este campo. Se comenzará explicando las técnicas y conceptos preliminares que nos han ayudado a obtener baselines y otros resultados para comparar los proporcionados por nuestros experimentos. Tras esto, se detallan los métodos que actualmente proporcionan los mejores resultados en clasificación de la polaridad a nivel monodominio y transdominio.

2.1. Técnicas y conceptos preliminares

Antes de adentrarse en la explicación de los diferentes métodos que conforman el estado del arte para clasificación de la polaridad a nivel mono y transdominio, en este primer apartado se explicarán algunas técnicas y conceptos básicos que se utilizarán más adelante para obtener unos resultados de referencia o baselines sobre los que se compararán los resultados de los experimentos proporcionados por nuestro método propuesto.

2.1.1. Técnicas de representación de texto

Cualquier algoritmo de aprendizaje automático trabaja con datos numéricos y, la mayoría de ellos, manejan entradas y salidas de datos de una longitud definida. Trabajar con texto nos plantea dos problemas: por un lado, los algoritmos no pueden tratar texto bruto y es necesario su conversión a una representación numérica manteniendo su significado. Por otro, los textos tienen una longitud variable y es raro que dos reviews coincidan en tamaño. A continuación, se detallan dos técnicas que nos permiten solucionar estos problemas. Todas estas técnicas se basan en el modelo de espacio vectorial [36]. Este modelo representa texto de una manera formal mediante el uso de vectores en un espacio lineal multidimensional.

Bolsa de palabras

El modelo de bolsa de palabras (*Bag-of-Words* o BoW) es una de las formas más intuitivas de representación vectorial numérica de texto. Este modelo viene definido por dos características clave: un vocabulario de palabras conocidas y una medida de la presencia de dichas palabras en el texto. Los documentos se representan con un vector, donde cada posición hace referencia a una palabra del vocabulario y su contenido a la presencia de dicha palabra en el documento. El vocabulario de palabras conocidas puede estar formado por palabras propiamente dichas o por conjuntos de n-gramas. Éste definirá la longitud del vector de representación.

Existen diferentes técnicas de medida de presencia de palabras. La más simple consiste en simplemente indicar si la palabra aparece o no en el documento, lo que daría lugar a un vector de unos y ceros. Otra medida algo más desarrollada consiste en llevar una cuenta de la cantidad de veces que aparece el término en el documento (dando lugar a un vector de enteros positivos). Por último, la técnica tf-idf (*term frequency-inverse document fre-*

quency) [34][35] viene definida por la siguiente fórmula:

$$\text{tf-idf}(w) = \text{tf}(w)N/\text{n}(w)$$

donde $\text{tf}(w)$ es el número de veces que el término w ocurre en el documento d, N es el número total de documentos y $\text{n}(w)$ es el número de documentos que contienen w.

Está técnica presenta una serie de limitaciones. Hay que tener especial cuidado para escoger un vocabulario representativo limitado, ya que su tamaño influirá en la longitud del vector de representación y, por tanto, en el tiempo de cómputo y el manejo de memoria del proceso. Normalmente, para restringir el tamaño del vocabulario y, por tanto, el tamaño del vector de representación, se suelen coger las N palabras más frecuentes en el vocabulario.

Además, se produce una pérdida de información al no conservarse el orden de las palabras. Esto puede paliarse, en parte, gracias al uso de combinación de n-gramas. Esto se explicará con más detenimiento en el apartado 2.1.2.

Representaciones distribuidas de palabras

Otra técnica de representación de texto que ha ganado popularidad en los últimos años es la de representaciones distribuidas de palabras o, más comúnmente denominada, *Word Embeddings* [24][20][5]. Ésta consiste en la asignación de un vector de números reales a cada palabra del vocabulario. Transforma el espacio con una dimensión por palabra antes mencionado a un espacio vectorial continuo con menos dimensiones. Habitualmente, esta representación trata de preservar la similaridad contextual que existe entre diferentes palabras, de manera que, las palabras que normalmente aparecen cercanas en un texto, obtienen representaciones que también son cercanas en el espacio vectorial. Esto se conseguiría entrenando algoritmos de aprendizaje automático para hacer predicciones basadas en las palabras

y sus contextos.

En el artículo Mikolov et al. (2013b) [25], los autores presentan una serie de arquitecturas para conseguir esta representación de manera eficiente, dando lugar al toolkit Word2Vec, [1] uno de los más empleados en los últimos estudios. Estas arquitecturas son el *continuous bag of Words model* (CBOW) y el *continuous skip-gram model*.

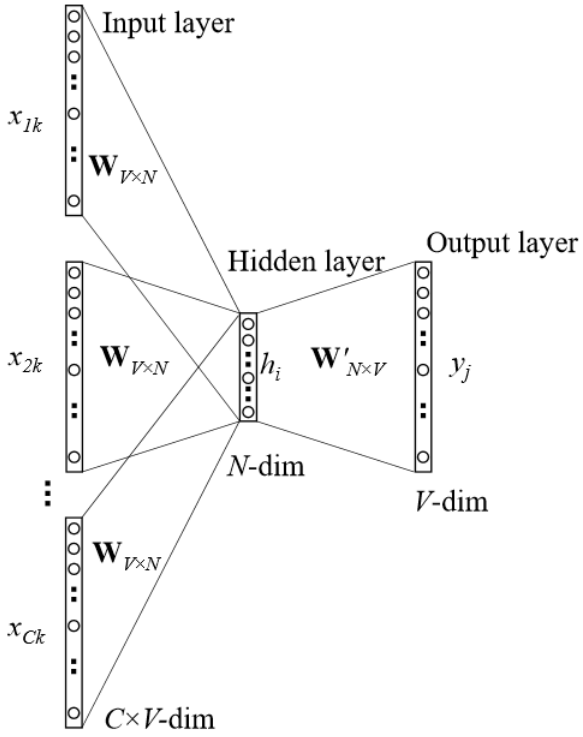

Figura 2.1: Arquitectura CBOW [33]

Para la arquitectura CBOW, se diseña el modelo para predecir una palabra (y_j) dado su contexto. En la figura 2.1 puede verse una representación gráfica del modelo. En primer lugar, se define un tamaño de ventana C,

[1] https://code.google.com/archive/p/word2vec/

siendo éste el número de palabras que se tendrán en cuenta como contexto de la palabra central. Partimos de la entrada $x_1, ..., x_C$, donde cada elemento x_i es la representación *one-hot* de esa palabra. El modelo necesita aprender los pesos correspondientes a dos matrices: $W \in \mathbb{R}^{V \times N}$ y $W' \in \mathbb{R}^{N \times V}$ donde N es la dimensión que tendrán los *embeddings* calculados y V la longitud del vocabulario. Obtenemos los vectores codificados de entrada por cada vector *one-hot* utilizando la matriz W y se calcula el vector h como la media de estos vectores:

$$h = \frac{1}{C} W^T (x_1 + x_2 + ... + x_C)$$

Obtenemos un vector de *scores* z tal que: $z = h^T W'$ y transformamos este vector en uno de probabilidades aplicando la función *hierarchical softmax* o la función *negative sampling* para calcular el error de predicción de y_j dado el contexto. Estos dos métodos se utilizan para poder estimar el error de la representación con respecto a una pequeña colección en lugar de la clásica normalización del *softmax* que hace que el coste computacional dependa de V. Puede verse una explicación más detallada de estas técnicas en [25] y [24].

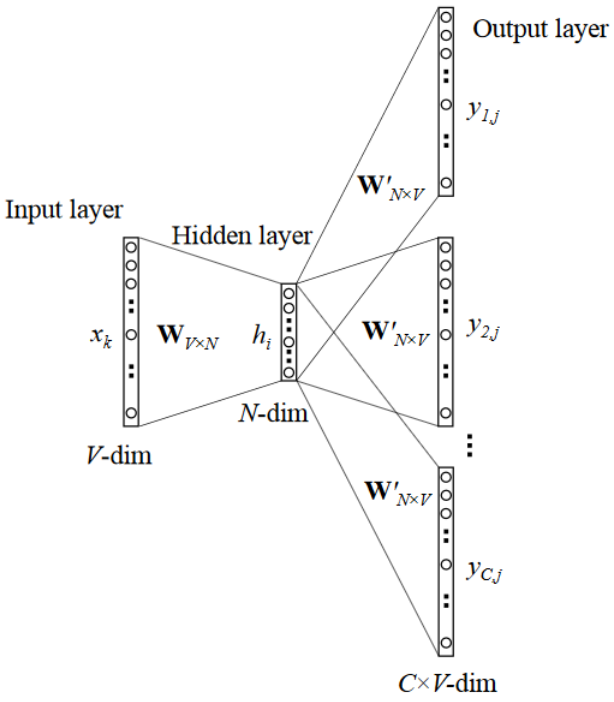

Figura 2.2: Arquitectura Skip-gram [33]

La arquitectura skip-gram es muy parecida a la ya mencionada CBOW. Su principal diferencia reside en que, en lugar de tratar de predecir una palabra a partir de su contexto, se entrena el modelo para predecir el contexto dada la palabra. Puede verse una representación gráfica del modelo en 2.2. Dada la representación *one-hot* de la palabra central x_k su codificación viene representada por la fila k-ésima de la matriz W, sin necesidad de calcular la media en este caso:

$$h = W^T x_k$$

Se calculan los C vectores de *scores* como $z_j = h^T W'$ y se transforman en vectores de probabilidad con las mismas técnicas mencionadas para el caso anterior.

Recientemente, en Bojanowski et al. (2016) [5] se presenta el método empleado en la biblioteca FastText. [2] Éste es una extensión del modelo empleado en Word2Vec pero, a diferencia de éste, FastText trata cada palabra como una composición de *n*-gramas de caracteres. El vector para cada palabra se obtiene con la suma de los vectores de sus *n*-gramas. Por este motivo, genera mejores *embeddings* para palabras poco frecuentes y puede construir *embeddings* de palabras no vistas durante el entrenamiento a partir de sus n-gramas de caracteres.

- **Representación de un documento con Word Embeddings**

Estas técnicas mencionadas, nos proporcionarán una representación vectorial para cada palabra del vocabulario, pero no una representación del documento en sí. Para obtener una representación vectorial del documento a partir de estos *embeddings*, existen diferentes técnicas. Una de ellas podría ser la concatenación de un número determinado de vectores, descartando las últimas palabras para los documentos más largos o rellenando con ceros los de insuficiente contenido. No obstante, hay estudios que demuestran que en textos cortos la mejor opción puede ser una media de los vectores que representan cada palabra del texto o una concatenación del vector mínimo y máximo [10].

2.1.2. Enriqueciendo las técnicas de representación de texto: Uso de *n*-gramas

Como se ha visto en el punto 2.1.1, existen técnicas de representación que no proporcionan información sobre el orden de las palabras dentro del documento. Para paliar parte de este problema, se puede hacer uso de *n*-gramas. Un *n*-grama es una subsecuencia de *n* elementos extraídos de una secuencia de texto. Estos elementos pueden ser tanto palabras como

[2] https://fasttext.cc/

caracteres. A los n-gramas de tamaño 1, 2, 3 y 4 se les denomina unigramas, bigramas, trigramas y cuatrigramas, respectivamente. Para todo tipo de longitudes se puede utilizar la nomenclatura n-gramas, por ejemplo: 5-gramas, 6-gramas, etc.

A continuación se muestra un ejemplo de extracción de unigramas, bigramas y trigramas de palabras para la frase 'El gato dijo miau':

- unigramas = ['El', 'gato', 'dijo', 'miau']
- bigramas = ['El gato', 'gato dijo', 'dijo miau']
- trigramas = ['El gato dijo', 'gato dijo miau']

Para la misma frase, se muestra un ejemplo de extracción de 6-gramas de caracteres:

- 6-gramas = ['El gat', 'l gato', ' gato ', 'gato d', 'ato di', 'to dij', 'o dijo', ' dijo ', 'dijo m', 'ijo mi', 'jo mia', 'o miau']

La elección de dividir por palabras o por caracteres dependerá del corpus y del método de clasificación que se vaya a emplear. En la práctica, es más común hacerlo por palabras y utilizar una longitud de n-gramas pequeña (entre 1 y 3). Una técnica que suele proporcionar mejores resultados es la combinación de diferentes longitudes [17][14].

2.1.3. Técnicas de clasificación

Una vez tenemos nuestro texto representado de forma numérica, necesitaremos seleccionar un algoritmo de clasificación adecuado para conseguir los primeros resultados de partida. Puesto que trabajaremos con datasets ya etiquetados, el algoritmo será de aprendizaje supervisado. Hemos seleccionado las máquinas de soporte vectorial [9] en su versión lineal por sus buenos resultados en el pasado en tareas de clasificación de texto [18].

Máquinas de Soporte Vectorial

Las máquinas de soporte vectorial, abreviado SVM por sus siglas en inglés (*Support Vector Machines*), es, en su formulación básica, un clasificador binario y lineal. Se considera de margen máximo, ya que busca el hiperplano que separe de forma óptima los puntos de una clase de los puntos de otra, ofreciendo la máxima distancia posible con los puntos más cercanos al mismo (vectores de soporte) (ver figura 2.3).

Dado un conjunto de entrenamiento $X = \{(x_1, y_1), ..., (x_n, y_n)\}$ donde $x \epsilon \mathbb{R}^d$ e $y_i \epsilon \{-1, +1\}$, el objetivo es encontrar w y b pertenecientes al hiperplano solución para aplicar la función $w^T x_i + b = 0$, donde w^T es el vector ortogonal al hiperplano y donde b es un umbral. La función de optimización del problema sería:

$$\arg\min_{w,b} \frac{1}{2}||w||^2 \text{ sujeto a } y_i(w\dot{x}_i + b) \geq 1, 1 \leq i \leq n$$

Introduciendo multiplicadores de Lagrange α, el problema puede expresarse como:

$$\arg\min_{w,b} \max_{\alpha \geq 0} \{\frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i [y_i(w\dot{x}_i - b) - 1]\}$$

Para la separación de conjuntos que no son linealmente separables pueden adoptarse algunas técnicas. Se puede relajar la condición de margen, lo que hace que se permita que algunas muestras no se clasifiquen correctamente. También se puede optar por transformar el espacio de características en otro de dimensión superior que sí sea linealmente separable (*kernel trick* [38]).

2.2. Métodos utilizados en clasificación monodominio

Como se ha mencionado en el apartado 1, la clasificación monodominio hace referencia a la tarea de clasificación de polaridad más común. Utiliza

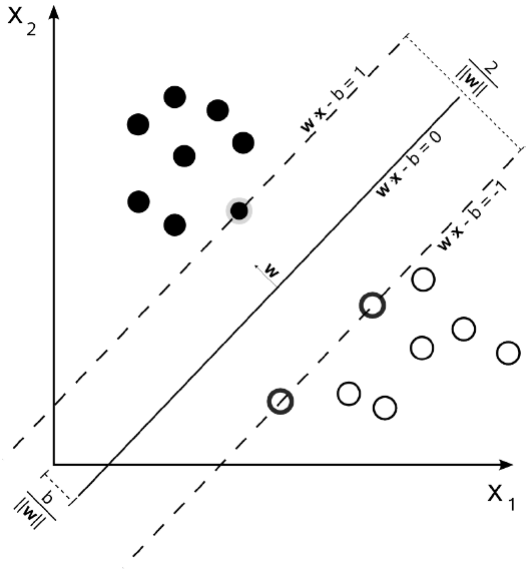

Figura 2.3: Hiperplano que maximiza la distancia entre las dos clases.

textos del mismo dominio tanto para la fase de entrenamiento como para la de testeo. En esta sección, se detalla uno de los métodos que, habiéndose evaluado con el mismo corpus que se utilizará para evaluar nuestro método, ha conseguido proporcionar los mejores resultados a día de hoy para el problema de la clasificación monodominio.

2.2.1. Clasificación lineal ponderada por la confianza

La clasificación lineal ponderada por la confianza [11], abreviado CWL por las siglas de Confidence-Weighted Linear Classification en inglés, es un método de aprendizaje online propuesto por Dredze et al. en 2008. Los autores observaron que las características más raras o poco frecuentes estaban ligadas a empobrecer el entrenamiento. Ilustran el caso con un ejemplo para el problema de clasificación de la polaridad. Teniendo en cuenta la review positiva "Me gusta este autor", una actualización del algoritmo online incrementaría el peso tanto para "gusta" como para "autor". Puesto

que las dos son palabras muy comunes, después de unas cuantas iteraciones, el algoritmo convergería asignando un peso positivo para "gusta" y un peso cercano a cero para "autor". Tengamos ahora en consideración la review "Me gusta este autor, pero el libro es tedioso". A pesar de que "tedioso" podría ser una característica discriminante, el algoritmo lo penalizaría y decrementaría el peso incorrectamente para "gusta" y no aportaría suficiente peso negativo a "tedioso".

Para solucionar este problema, los autores proponen actualizar más agresivamente los pesos para las características menos frecuentes. Este método modela la incertidumbre de los pesos de las características manteniendo una distribución normal sobre el vector de pesos de un clasificador lineal. La media representa el vector promedio de pesos y la desviación estándar captura la incertidumbre de los pesos. Esta incertidumbre será la que determine cómo de agresiva será la actualización de los pesos. El algoritmo se inicializa con una media μ y matriz de covarianza Σ proporcionadas por el usuario. Cada vez que entra una nueva muestra de entrenamiento, los parámetros se actualizan de modo que la divergencia de Kullback-Leibler [19] entre la actual distribución gaussiana y la nueva distribución sea mínima, a la vez que la probabilidad de clasificar correctamente la muestra sea mayor a un umbral determinado η:

$$(\mu_{i+1}, \Sigma_{i+1}) = \text{mín}\, D_{KL}(\mathcal{N}(\mu, \Sigma), \mathcal{N}(\mu_i, \Sigma_i)) \text{ s.t. } Pr[y_i(w \cdot x_i) \geq 0] \geq \eta]$$

Este modelo se probó en diferentes tareas de NLP, con muy buenos resultados para el Multi-Domain Sentiment Dataset.

2.3. Métodos utilizados en clasificación tansdominio

La clasificación de la polaridad transdominio se centra en utilizar un dominio para la fase de entrenamiento (dominio fuente) diferente al que se utilizará en la fase de testeo (dominio objetivo). La expresión del sentimiento se realiza de manera diferente dependiendo del dominio sobre el

que se esté opinando. Palabras como luminoso o duradero pueden estar reflejando un sentimiento positivo si se habla sobre un producto electrónico que no tiene por qué extenderse a otros dominios como, por ejemplo, si se está opinando sobre un libro. Por este motivo, el estudio de la clasificación transdominio adquiere especial interés para casos en los que sea difícil conseguir muestras etiquetadas del dominio objetivo. A continuación, se explica uno de los métodos que ha sido evaluado con el Multi-Domain Sentiment Dataset para esta tarea y ha conseguido proporcionar buenos resultados.

2.3.1. Aprendizaje de correspondencia estructural

Blitzer et al. [3] fueron los primeros en porporcionar resultados en clasificación CD utilizando el Multi-Domain Sentiment Dataset. Para ello proponen los métodos de aprendizaje de correspondencia estructural o SCL (de sus siglas en inglés: *Structural Cosrrespondence Learning*) y su variante utilizando información mutua, SCL-MI (*Structural Correspondence Learning with Mutual Information*).

El algoritmo SCL parte de datos etiquetados en el dominio fuente y datos sin etiquetar tanto del dominio fuente como del objetivo. En primer lugar, SCL elige un conjunto de m características pivote (unigramas y bigramas) que aparecen con más frecuencia en los dos dominios. Después, modela la correlación entre las características pivote y todas las demás entrenando predictores lineares de pivotes para predecir la aparición de cada uno en las muestras no etiquetadas de los dos dominios. El l-ésimo predictor de pivote se caracteriza por su vector de pesos w_l; las entradas positivas en ese vector quieren decir que una característica no pivote está altamente correlacionada con ese pivote. La matriz consistente en todos los vectores de pesos se procesa mediante la descomposición de valores singulares o SVD (*singular value decomposition* [16]) para reducir dimensiones.

Selección de pivotes con información mutua

La eficacia de SCL depende de la selección de pivotes. En un estudio previo, Blitzer et al. [4] muestran como la selección de las palabras más frecuentes en ambos dominios dan buenos resultados para el problema de la categorización gramatical o PoS (del inglés, *part-of-speech*), ya que éstas suelen ser determinantes, preposiciones, etc. Para el caso del análisis del sentimiento, este tipo de palabras no son una buena opción. Por ello, seleccionan las características que tengan información mutua con la etiqueta fuente. Por ejemplo: "Excellent" es una palabra que aparece en el dominio "Books" y "DVD" y que tiene un alto MI (*Mutual Information*) con la etiqueta de sentimiento positivo.

2.4. Métodos utilizados tanto en clasificación mono como transdominio

En este apartado, se explican otros métodos que han sido evaluados con el mismo dataset ya nombrado para las dos tareas que nos interesan: clasificación de la polaridad en mono y transdominio y que han proporcionado los mejores resultados para este problema.

2.4.1. Tesauros sensibles al sentimiento

Bollegala et al. proponen el método SST [6] (de sus siglas en inglés, *Sentiment Sensitive Thesaurus*) que se basa en crear un tesauro que agrupa diferentes palabras que expresan el mismo sentimiento, utilizando unigramas y bigramas como representación. El método hace uso tanto de reviews etiquetadas en el dominio fuente, como de reviews sin etiquetar del dominio fuente y del dominio objetivo.

Primero, dividen todas las reviews (etiquetadas y no etiquetadas) en

sentence	Excellent and broad survey of the development of civilization.
POS tags	Excellent/JJ and/CC broad/JJ survey/NN1 of/IO the/AT development/NN1 of/IO civilization/NN1
lexical elements (unigrams)	excellent, broad, survey, development, civilization
lexical elements (bigrams)	excellent+broad, broad+survey, survey+development, development+civilization
sentiment elements	excellent*P, broad*P, survey*P, development*P, civilization*P, excellent+broad*P, broad+survey*P, survey+development*P, development+civilization*P

Cuadro 2.1: Ejemplo de generación de elementos léxicos y de sentimiento en SST para un frase de review.

frases individuales. Realizan un etiquetado PoS, para quedarse únicamente con nombres, verbos, adjetivos y adverbios, y lematizan los elementos. Para cada review etiquetada de dominio fuente, crean elementos de sentimiento, añadiendo la etiqueta de la review a cada elemento léxico perteneciente a ella (*P para las positivas y *N para las negativas). En la tabla 2.1 se muestra un ejemplo.

Cada elemento léxico o de sentimiento u se representa con un vector de características $\mathbf{u}$, donde cada elemento léxico o de sentimiento w que coocurre con u en una frase de review contribuye con una característica al vector $\mathbf{u}$. El vector $\mathbf{u}$ puede verse como una representación compacta de la distribución de un elemento u sobre el conjunto de elementos que coocurren con u en las reviews. El valor de la característica w en el vector $\mathbf{u}$ se denota $f(\mathbf{u}, w)$ y se calcula de la siguiente manera:

$$f(\mathbf{u}, w) = log(\frac{\frac{c(u,w)}{N}}{\frac{\sum_{i=1}^{n} c(i,w)}{N} \times \frac{\sum_{j=1}^{n} c(u,j)}{N}})$$

Donde $c(u, w)$ denota el número de frases de reviews en las que el elemento léxico u y la característica w coocurren, n y m respectivamente denotan el número total de elementos léxicos y el número total de características, y $N = \sum_{i=1}^{n} \sum_{j=1}^{m} c(i, j)$.

Seguidamente, para dos elementos léxicos o de sentimiento u y v (representados por los vectores $\mathbf{u}$ y $\mathbf{v}$), se calcula la relación $\tau(u, v)$ del elemento

v con el elemento u como sigue:

$$\tau(u,v) = \frac{\sum_{w \in \{x|f(\mathbf{v},x)>0\}} f(\mathbf{u},w)}{\sum_{w \in \{x|f(\mathbf{u},x)>0\}} f(\mathbf{u},w)}$$

donde $\{x|f(\mathbf{v},x) > 0\}$ es el conjunto de características que coocurren con v. Si no hay características que coocurran con u y v, su relación será 0. Si todas las características que coocurren con u, lo hacen con v, su relación será el máximo valor 1. Con esta medida de relación, se construye el tesauro.

Con el tesauro ya calculado y con cada review modelada como un BoW, se lleva a cabo una fase de expansión de características. Dada una review d representada con el vector $\mathbf{d}$ usando el conjunto $\{w_1, ..., w_N\}$ (donde los elementos w_i son los unigramas y bigramas que aparecen en la review d), se buscan candidatos para expandir dicho vector definiendo un ranking con las entradas del tesauro. Para cada entrada se calcula su score de la siguiente manera:

$$\text{score}(u_i, \mathbf{d}) = \frac{\sum_{j=1}^{N} d_j \tau(w_j, u_i)}{\sum_{l=1}^{N} d_l}$$

Con los k valores más altos se expande el vector $\mathbf{d}$. Por último, como algoritmo de clasificación utilizan regresión logística.

2.4.2. Meta-clasificador mejorado con conocimiento

En [12] los autores proponen el método Knowledge-Enhanced Meta classifier (KE-Meta), que proporciona resultados prometedores tanto en mono como en transdominio. A diferencia del método presentado en el apartado anterior, para el problema de la clasificación de la polaridad transdominio, no se necesita realizar una adaptación del dominio. Por tanto, no son necesarias muestras sin etiquetar del dominio objetivo y el proceso de aprendizaje se lleva a cabo únicamente con las muestras etiquetadas del dominio fuente.

El método combina diferentes enfoques clásicos (BoW, n-gramas, clasificadores basados en recursos léxicos), añadiendo además otros clasificadores basados en conocimiento.

Desambiguación de significados y expansión del vocabulario

En primer lugar, los autores utilizan la red semántica BabelNet [3] [26] para llevar a cabo la desambiguación de las palabras o WSD (del inglés, *word sense disambiguation*), es decir, identificar el sentido que se está empleando de una palabra cuando ésta es polisémica. Este proceso se lleva a cabo en cinco pasos:

1. Etiquetado PoS y lematización: se obtiene la lista de tuplas (lemma, tag) T, quedándose únicamente con los adverbios, nombres, verbos y adjetivos.

2. Creación de un grafo con los conceptos iniciales: Se crea un grafo de conocimiento inicialmente vacío $G = (V, E)$. Se rellena el conjunto de vértices V con el conjunto $S_k = \bigcup_{t \in T} Synsets_L(t)$, el cual es el conjunto de sinónimos en BabelNet que contenga cualquier tupla de T para el lenguaje L.

3. Se crea el grafo de conocimiento G buscando en BabelNet el conjunto de rutas P que conectan los pares de synsets en V.

4. Se añaden los pesos a los conceptos y a las relaciones semánticas del grafo G. Para ponderar las relaciones se utilizan los pesos originales de BabelNet, los cuales son una medida del grado de relación entre los synsets conectados por cada arista. Para ponderar los conceptos se utiliza su propia centralidad.

5. Selección de las desambiguaciones: para cada tupla (lemma, tag) $t \in T$ se selecciona de Babelnet el conjunto de synsets que contienen t

[3] http://babelnet.org

y se selecciona como desambiguación t_{WSD} la de mayor puntuación: $t_{WSD} = argmax_{s \in S_t} score(s)$

Con el grafo G calculado y el conjunto de desambiguaciones S_{WSD}, se borran todas las rutas que no están en el conjunto de desambiguaciones. Después, se expande el vocabulario incluyendo los conceptos intermedios en las rutas que han quedado.

Proceso de clasificación

Este proceso de clasificación hace uso de ocho métodos diferentes:

1. Clasificador tf-idf con SVM.
2. Clasificador tf-idf combinando unigramas, bigramas y trigramas con SVM.
3. Clasificador basado en recursos léxicos: utilizan el ML-Senticon[4] que asocia tres puntuaciones (negatividad, positividad y objetividad) a cada synset en combinación con un clasificador basado en árboles de decisión.
4. Cuatro clasificadores basados en WSD: se genera un clasificador independiente para cada etiqueta PoS (adverbios, verbos, nombres y adjetivos), utilizando un BoW de presencia en combinación con SVM.
5. Clasificador basado en expansión del vocabulario: utilizando la expansión de vocabulario explicada anteriormente, se representa cada documento como un BoW de presencia y se utiliza el clasificador SVM.

Estos ocho clasificadores se combinan utilizando el método de meta-aprendizaje generalización apilada (*Stacked Generalization*) [40]. En un primer nivel, se

[4] http://timm.ujaen.es/recursos/ml-senticon/

obtienen las probabilidades de clase de los clasificadores. Un segundo nivel utiliza estas anotaciones para obtener una decisión final.

Capítulo 3

String Kernels

En este apartado se explica el método propuesto para la resolución de las cuestiones planteadas en el punto 1.1. Se comenzará dando una explicación detallada del objetivo de los SK y su implementación. También se introducirá el modelo para procesar los SK mencionado anteriormente, KDA.

3.1. Introducción a los String Kernels

Aunque comúnmente los algoritmos de representación de texto en forma numérica para tareas de NLP suelen funcionar a nivel de palabra, son ya numerosos los estudios que han demostrado efectividad y proporcionado buenos resultados trabajando a nivel de carácter [21][37][28][17].

Las funciones de kernels otorgan a los métodos de kernels el poder para manejar de forma natural datos de entrada que no se representan a priori en forma de vectores numéricos como, por ejemplo, cadenas de caracteres o imágenes. Estas funciones capturan la similitud entre objetos en un dominio específico y pueden ser cualquier función definida en este dominio que sea simétrica y definida positiva.

Los SK son funciones de kernels que trabajan sobre secuencias que no tienen una longitud fija definida (por ejemplo, secuencias de caracteres o secuencias genéticas). A continuación se van a detallar diferentes funciones de SK según la implementación y formulación vista en [17].

Una de las formas más intuitiva de medir la similitud entre dos cadenas es contar cuántas subcadenas de longitud p tienen las dos cadenas en común. Esta función se denomina kernel *spectrum*. Más formalmente, para dos cadenas pertenecientes a un alfabeto Σ, $s, t \in \Sigma^*$, la función kernel *spectrum* se define como:

$$K_p(s,t) = \sum_{v \in \Sigma^p} \text{num}_v(s) \cdot \text{num}_v(t)$$

donde $\text{num}_v(s)$ es el número de ocurrencias de la cadena v como subcadena en s. El mapa de características definido por esta función asocia un vector de dimensión $|\Sigma|^p$ que contiene el histograma de frecuencias de todas sus subcadenas de longitud p (p-gramas) con cada cadena.

El kernel *presence* modifica el mapa de características del kernel anterior para asociarle un vector, también de longitud $|\Sigma|^p$, que contiene los bits de presencia (en lugar de las frecuencias) de todas sus subcadenas de longitud p con cada cadena. Por tanto, el kernel de bits de presencia de p-gramas de caracteres se obtiene de la siguiente manera:

$$k_p^{0/1}(s,t) = \sum_{c \in \Sigma^p} \text{in}_v(s) \cdot \text{in}_v(t)$$

donde $\text{in}_v(s)$ es 1 si la cadena v ocurre como subcadena de s y 0 si no.

El último de los kernels que se emplea en este método es el kernel *intersection*. Aunque ya se había utilizado con éxito en visión por computador, se utiliza por primera vez en una tarea de NLP en [17]. Se define de la siguiente manera:

$$k_p^{\cap}(s,t) = \sum_{v \in \Sigma^p} \min\{\text{num}_v(s), \text{num}_v(t)\}$$

donde $\text{num}_v(s)$ es el número de ocurrencias de la cadena v como subcadena en s.

Para el kernel *spectrum*, la frecuencia de un p-grama tiene una contribución muy significativa al valor final, ya que está teniendo en cuenta el producto de las frecuencias. Por otro lado, la frecuencia de un p-grama se ignora por completo en el kernel de presencia de bits (*presence*). Entre estos dos kernels, el kernel *intersection* proporcionaría unos valores más moderados. La siguiente relación de desigualdad matemática describe la relación entre los tres kernels:

$$k_p^{0/1}(s,t) \leq k_p^{\cap}(s,t) \leq k_p(s,t)$$

Lo que se puede sacar en conclusión de estas implementaciones es que el kernel *intersection* asignará un valor alto a un p-grama cuando éste ocurra con frecuencia en las dos cadenas, ya que considera el mínimo de las dos frecuencias. Por otro lado, el kernel *spectrum* asignará valores altos incluso cuando el p-grama sólo ocurra con alta frecuencia en una de las dos cadenas. Por tanto, el kernel *intersection* captura cierta correlación entre las frecuencias de los p-gramas en ambas cadenas, lo que conlleva mayor sensibilidad a la similitud entre ellas.

Para asegurar una comparación justa entre los valores de estos kernels, se lleva a cabo un proceso de normalización:

$$\hat{k}_p(s,t) = \frac{k_p(s,t)}{\sqrt{k_p(s,s) \cdot k_p(t,t)}}$$

$$\hat{k}_p^{0/1}(s,t) = \frac{k_p^{0/1}(s,t)}{\sqrt{k_p^{0/1}(s,s) \cdot k_p^{0/1}(t,t)}}$$

$$\hat{k}_p^{\cap}(s,t) = \frac{k_p^{\cap}(s,t)}{\sqrt{k_p^{\cap}(s,s) \cdot k_p^{\cap}(t,t)}}$$

A continuación, se muestra el cálculo de String Kernels con un pequeño ejemplo. Dado el par de frases s = "Get the dress in that address" y t = "A blue dress on a dresser", y teniendo en cuenta una longitud de p-grama de 5:

- Se obtiene el conjunto de 5-gramas presentes en las dos frases: L^5 = ('e dre', ' dres', 'dress', 'ress ')
- Se calculan los valores para cada función kernel:
  - $K_5(s,t) = 1 \cdot 1 + 1 \cdot 2 + 2 \cdot 2 + 1 \cdot 1 = 8$
  - $K_5^{0/1}(s,t) = 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 = 4$
  - $K_5^{\cap}(s,t) = 1 + 1 + 2 + 1 = 5$

Como se puede comprobar, la relación de desigualdad antes mencionada se cumple en este caso.

Los SK meten implícitamente los textos en un espacio de características de alta dimensión. Tras esto, hay que seleccionar un algoritmo de aprendizaje basado en kernels que asigne implícitamente un peso a cada característica, seleccionando así las características que son importantes para la tarea de discriminación. Para la resolución de este problema, se usará el algoritmo Kernel Discriminant Analysis (KDA).

3.2. Kernel Discriminant Analysis

KDA [23][2] es la versión kernelizada del algoritmo Linear Discriminant Analysis (LDA) [13]. Éste es un método de reducción de la dimensionalidad. El LDA se lleva a cabo en el espacio de entrada original y el KDA en el espacio de Hilbert con kernel reproductor o RKHS (del inglés, *reproducing kernel Hilbert space*).

El algoritmo LDA busca direcciones en las que los puntos de datos de diferentes clases están lo más alejados entre sí, a la vez que requiere que los puntos de datos de la misma clase estén cerca unos de otros. A continuación se explica el algoritmo utilizando la nomenclatura utilizada en Cai et al. (2011) [8].

Supongamos que tenemos un conjunto de m muestras $x_1, x_2, ..., x_m \in \mathbb{R}^n$, pertenecientes a c clases. La función objetivo de LDA es la siguiente:

$$\mathbf{a}_{opt} = \arg\max \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T S_w \mathbf{a}}$$

$$S_b = \sum_{k=1}^{c} m_k (\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu})(\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu})^T$$

$$S_w = \sum_{k=1}^{c} (\sum_{i=1}^{m_k} (\boldsymbol{x}_i^{(k)} - \boldsymbol{\mu}^{(k)})(\boldsymbol{x}_i^{(k)} - \boldsymbol{\mu}^{(k)})^T)$$

donde $\boldsymbol{\mu}$ es el centroide global, m_k es el número de muestras en la k-ésima clase, y $\boldsymbol{x}_i^{(k)}$ es la i-ésima muestra en la k-ésima clase. S_w es la matriz de dispersión intraclase y S_b la matriz de dispersión interclases. Se define la matriz de dispersión total como:

$$S_t = \sum_{i=1}^{m} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T$$

que cumple la relación $S_t = S_b + S_w$. La función objetivo mencionada anteriormente es equivalente a:

$$\mathbf{a}_{opt} = \arg\max \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T S_t \mathbf{a}}$$

Los $\mathbf{a}$ óptimos son los vectores propios correspondientes a los valores propios distintos de cero del problema propio:

$$S_b \mathbf{a} = \lambda S_t \mathbf{a}$$

Como el rango de S_b está limitado por $c - 1$, existen como mucho $c - 1$ vectores propios correspondientes a valores propios distintos de cero.

Para extender LDA al caso no lineal, consideramos el problema en un espacio de características $\mathcal{F}$ inducido por un mapeo no lineal:

$$\phi : \mathbb{R}^n \to \mathcal{F}$$

Para una correcta elección ϕ, se puede definir un producto interno $\langle , \rangle$ en $\mathcal{F}$ el cual crea un RKHS. Más concretamente:

$$\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle = \mathcal{K}(\boldsymbol{x}, \boldsymbol{y})$$

donde $\mathcal{K}(.,.)$ es una función kernel.

Siendo S_b^ϕ,S_w^ϕ y S_t^ϕ las matrices de dispersión intraclase, interclase y total, respectivamente, tenemos:

$$S_b^\phi = \sum_{k=1}^{c} m_k(\boldsymbol{\mu}_\phi^{(k)} - \boldsymbol{\mu}_\phi)(\boldsymbol{\mu}_\phi^{(k)} - \boldsymbol{\mu}_\phi)^T$$

$$S_w^\phi = \sum_{k=1}^{c}(\sum_{i=1}^{m_k}(\phi(\boldsymbol{x}_i^{(k)}) - \boldsymbol{\mu}_\phi^{(k)})(\phi(\boldsymbol{x}_i^{(k)}) - \boldsymbol{\mu}_\phi^{(k)})^T)$$

$$S_t^\phi = \sum_{i=1}^{m}(\phi(\boldsymbol{x}_i) - \boldsymbol{\mu}_\phi)(\phi(\boldsymbol{x}_i) - \boldsymbol{\mu}_\phi)^T$$

donde $\mu_\phi^{(k)}$ y μ_ϕ son los centroides de la k-ésima clase y el centroide global, respectivamente, en el espacio de características.

Siendo $\boldsymbol{\nu}$ la función que proyecta en el espacio de características, la función objetivo en este espacio es:

$$\boldsymbol{\nu}_{opt} = \arg\max \frac{\boldsymbol{\nu}^T S_b^\phi \boldsymbol{\nu}}{\boldsymbol{\nu}^T S_t^\phi \boldsymbol{\nu}}$$

la cual puede ser calculada por el problema propio:

$$S_b^\phi \boldsymbol{\nu} = \lambda S_t^\phi \boldsymbol{\nu}$$

Debido a que los vectores propios son combinaciones lineales de $\phi(\boldsymbol{x}_i)$, existen coeficientes α_i tales que:

$$\boldsymbol{\nu} = \sum_{i=1}^{m} \alpha_i \phi(\boldsymbol{x}_i)$$

Dado $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_m]^T$, se ha demostrado [2] que la ecuación de valores óptimos declarada anteriormente es equivalente a:

$$\boldsymbol{\alpha}_{opt} = \arg\max \frac{\boldsymbol{\alpha}^T KWK\boldsymbol{\alpha}}{\boldsymbol{\alpha}^T KK\boldsymbol{\alpha}}$$

y su correspondiente problema propio:

$$KWK\boldsymbol{\alpha} = \lambda KK\boldsymbol{\alpha}$$

donde K es la matriz kernel y W se define como:

$$W_{ij} = \begin{cases} 1/m_k, & \text{si } \boldsymbol{x_k} \text{ y } \boldsymbol{x_j} \text{ pertenecen ambas a la } k\text{-ésima clase.} \\ 0, & \text{en otro caso.} \end{cases}$$

Cada vector propio $\boldsymbol{\alpha}$ proporciona una función de proyección $\boldsymbol{\nu}$ en el espacio de características. Para un dato $\boldsymbol{x}$ de ejemplo tenemos:

$$\langle \boldsymbol{\nu}, \phi(\boldsymbol{x}) \rangle = \sum_{i=1}^{m} \alpha_i \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle = \sum_{i=1}^{m} \alpha_i \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}) = \boldsymbol{\alpha}^T K(:, \boldsymbol{x})$$

donde $K(:, \boldsymbol{x}) = [K(\boldsymbol{x}_1, \boldsymbol{x}), ..., K(\boldsymbol{x}_m, \boldsymbol{x})]^T$. Dada la matriz de transformación $\Theta = [\boldsymbol{\alpha_1}, ..., \boldsymbol{\alpha_{c-1}}]$ de dimensión $m \times (c-1)$, una muestra $\boldsymbol{x}$ puede ser transformada en el espacio de características como:

$$\boldsymbol{x} \to \boldsymbol{z} = \Theta^T K(:, (\boldsymbol{x}))$$

3.2.1. Clasificación utilizando KDA

Para la resolución de nuestro problema utilizamos la implementación de Baudat et al. (2000) [2] [1] para obtener la matriz de vectores propios Θ. Se sigue el método propuesto por Ionescu et al. (2014) [17] para clasificar con SK y KDA. Obtenemos las matrices de características calculando $Y = K\Theta$ e $Y_t = K_t\Theta$, donde K y K_t son los kernels de entrenamiento y test. Para cada clase c, creamos el prototipo Y_c como la media de todos los vectores de Y que corresponden a instancias de la clase c. Finalmente, clasificamos cada muestra de test identificando la clase del prototipo con menor error cuadrático medio entre $Y_t(i)$ e Y_c.

[1] http://www.cad.zju.edu.cn/home/dengcai/Data/DimensionReduction.html

Capítulo 4

Evaluación

En este capítulo se entrará en detalle en el diseño de los experimentos para la evaluación del modelo propuesto en el capítulo 3 y en el análisis y comparación de los resultados obtenidos, tratando de dar respuesta a las cuestiones planteadas en el capítulo 1. Para ello, el proceso de evaluación se dividirá en dos fases. En la primera, analizaremos los resultados de aplicar nuestro método propuesto al corpus Multi-Domain Sentiment Dataset. En la segunda trabajaremos sobre el corpus Cross Lingual Sentiment.

4.1. Multi-Domain Sentiment Dataset (v. 2.0)

4.1.1. Corpus

El Multi-Domain Sentiment Dataset (v. 2.0) es un corpus formado por diferentes reviews en inglés extraídas de la página Amazon.com en diferentes categorías de productos (dominios). Cada una de estas reviews posee en sus metadatos información acerca de la nota otorgada por el usuario (que va de 0 a 5 estrellas), el nombre y la localidad del usuario, el nombre del producto, el título y la fecha de la review, y el texto. Un subconjunto de las

reviews que poseen una valoración menor a 3 estrellas están etiquetadas como negativas y, otro subconjunto de reviews que tienen una valoración mayor a 3 estrellas están etiquetadas como positivas. Siguiendo la literatura, utilizamos para evaluar nuestro método únicamente los textos de las reviews de los dominios Books, DVDs, Electronics y Kitchen appliances y sus respectivas etiquetas de clase. Cada uno de estos dominios, proporciona 1000 reviews positivas y 1000 reviews negativas, lo que representa un total de 8000 reviews etiquetadas.

4.1.2. Metodología

Evaluamos nuestro método utilizando los kernels *presence* ($k_p^{0/1}$), *intersection* ($k_p^{\cap}$) y *spectrum* (k_p). Comparamos los resultados con los métodos explicados en el capítulo 2. Para el nivel monodominio, comparamos con los métodos KE-Meta, SST y CWL. Para el nivel transdominio, comparamos con los métodos KE-Meta, SST y SCL-MI. En los dos casos, comparamos también con los resultados proporcionados por un baseline que utiliza combinación de unigramas, bigramas y trigramas representados en un BoW con valores tf-idf y un algoritmo de clasificación SVM con kernel lineal (referenciado en las tablas como *word n-g*). La evaluación se lleva a cabo utilizando una validación cruzada estratificada de 10 particiones y, utilizamos la exactitud (*accuracy*) como métrica de evaluación.

Validación cruzada

Si entrenamos con los mismos datos que van a ser utilizados a la hora de evaluar el modelo, estamos cometiendo un error metodológico. El modelo ya ha visto esas muestras y puede predecirlas con exactitud, lo cual no estaría reflejando cuál sería su calidad a la hora de predecir muestras que nunca antes ha visto. A este problema se le denomina sobreajuste u *overfitting*. Para evitarlo, normalmente se reserva una parte de los datos

disponibles para evaluar el modelo una vez ya ha sido entrenado con los datos restantes.

Cuando el número de muestras totales es limitado, una forma de poder aprovechar al máximo todos los datos para entrenar el modelo y evaluarlo sin estar cometiendo este error, sería la validación cruzada de N particiones. Con esta técnica, el conjunto total de muestras se divide en N particiones de igual tamaño. De estos N conjuntos, uno se selecciona como partición de validación, y los $N-1$ restantes se utilizan en conjunto para entrenar el modelo. Este proceso se repite N veces para cada una de las particiones. La estimación final de su rendimiento será el promedio de los N resultados obtenidos. El hecho de que sea estratificada implica que la proporción de datos pertenecientes a cada clase debe ser similar a lo largo de todas las particiones.

Métrica de evaluación: Exactitud

La exactitud o *accuracy* es una de las métricas más sencillas e intuitivas para la evaluación de resultados. Consiste en la división entre el número de muestras etiquetadas correctamente y el número total de muestras. Más formalmente:

$$\text{accuracy}(y, \hat{y}) = \frac{\sum_{i=0}^{n-1} 1(\hat{y}_i = y_i)}{n}$$

donde n es el número total de muestras, $\hat{y}$ es el vector con las etiquetas de clase predichas e y es el vector con las etiquetas de clase reales.

4.1.3. Selección de parámetros

En primer lugar, se hizo un estudio para definir la mejor combinación de longitudes de n-gramas de caracteres. Para ello, se evaluó en primera instancia los resultados de aplicar longitudes individuales para definir los rangos en los que pueden obtenerse los mejores resultados.

Long. *n*-grama	Books	Dvd	Electronics	Kitchen	Media
2	69.70	70.70	74.30	74.90	72.40
3	78.30	80.20	80.30	81.30	80.03
4	82.30	82.40	83.70	83.30	82.93
5	83.10	**83.70**	84.90	85.20	84.23
6	**86.50**	83.50	85.00	85.40	**85.10**
7	83.80	83.10	85.40	**86.50**	84.70
8	83.00	83.10	**85.50**	85.70	84.33
9	82.80	81.90	84.70	84.80	83.55
10	82.00	81.10	84.00	84.90	83.00

Cuadro 4.1: Valores de accuracy (en %) obtenidos con la función kernel presence y un valor α de 0.2 con cada una de las longitudes de n-gramas establecidas.

Utilizando únicamente el kernel *presence* y un valor de α fijo de 0.2, se obtuvieron los resultados mostrados en la tabla 4.1. Como se puede apreciar, se empiezan a obtener resultados competentes a partir de una longitud de 4 y, para longitudes mayores que 9, dejan de apreciarse mejoras. Por tanto, definimos nuestro rango de evaluación para longitudes entre 4 y 9.

Comb. *n*-gramas	Books	Dvd	Electronics	Kitchen	Media
[4-9]	85.00	84.90	85.60	86.00	**85.38**
[4-8]	85.10	84.60	85.30	85.80	**85.20**
[4-7]	84.30	84.10	85.40	85.30	84.78
[4-6]	83.70	83.90	84.60	85.10	84.33
[4-5]	83.20	83.10	84.20	84.20	83.68
[5-9]	84.20	84.10	86.00	86.20	85.13
[5-8]	84.70	84.30	85.80	86.00	**85.20**
[5-7]	84.70	83.90	85.30	85.40	84.83
[5-6]	83.80	83.90	85.10	85.20	84.50
[6-9]	83.80	83.90	85.70	86.30	84.93
[6-8]	84.20	83.70	85.80	86.50	85.05
[6-7]	84.10	83.80	85.50	85.80	84.80
[7-9]	83.90	83.40	85.30	86.10	84.68
[7-8]	83.40	83.00	85.10	86.20	84.43
[8-9]	82.80	82.80	85.50	85.40	84.13

Cuadro 4.2: Valores de accuracy (en %) obtenidos con la función kernel presence y un valor α de 0.2 con cada una de las combinaciones de longitudes de n-gramas establecidas.

En la tabla 4.2 se muestran los resultados proporcionados para las diferentes combinaciones de longitudes de *n*-gramas. En promedio, se obtienen resultados similares para los rangos $[4-9]$, $[4-8]$ y $[5-8]$. Por comodidad y por la reducción de tiempo de cómputo que supone se elige el rango $[5-8]$.

		Max α Monodominio	Max α Transdominio
$\mathbf{k}_p^{\cap}$	**Books**	0.5	0.6
	DVDs	0.7	0.6
	Electronics	0.4	0.2
	Kitchen	0.5	0.8
$\mathbf{k}_p$	**Books**	0.2	0.7
	DVDs	0.5	0.7
	Electronics	0.3	0.01
	Kitchen	0.7	0.9
$\mathbf{k}_p^{0/1}$	**Books**	0.3	0.2
	DVDs	0.6	1.0
	Electronics	0.3	0.4
	Kitchen	0.5	0.9

Cuadro 4.3: Valores de α que proporcionan los mejores resultados en promedio para cada dominio y kernel.

Con el rango de n-gramas óptimo ya establecido, se procede al tuneo del parámetro α para el algoritmo KDA. Se definen las 10 particiones de entrenamiento y 10 particiones de test sobre las que se evaluará en un próximo paso el modelo. Para cada partición de entrenamiento, se obtiene de manera aleatoria y equilibrada dos subparticiones train y development con una proporción 80 %-20 %. Para cada partición, dominio y kernel, se evalúan valores de $\alpha \in [0{,}01, 1]$. El valor de alpha que proporciona el mejor resultado se utilizará luego para cada partición de test. En la tabla 4.3 se muestra el mejor valor de alpha en promedio para cada dominio y kernel. Puede verse la variación de los resultados obtenidos en la gráfica 4.1. Los parámetros ajustados en esta sección se utilizarán para el resto de la evaluación.

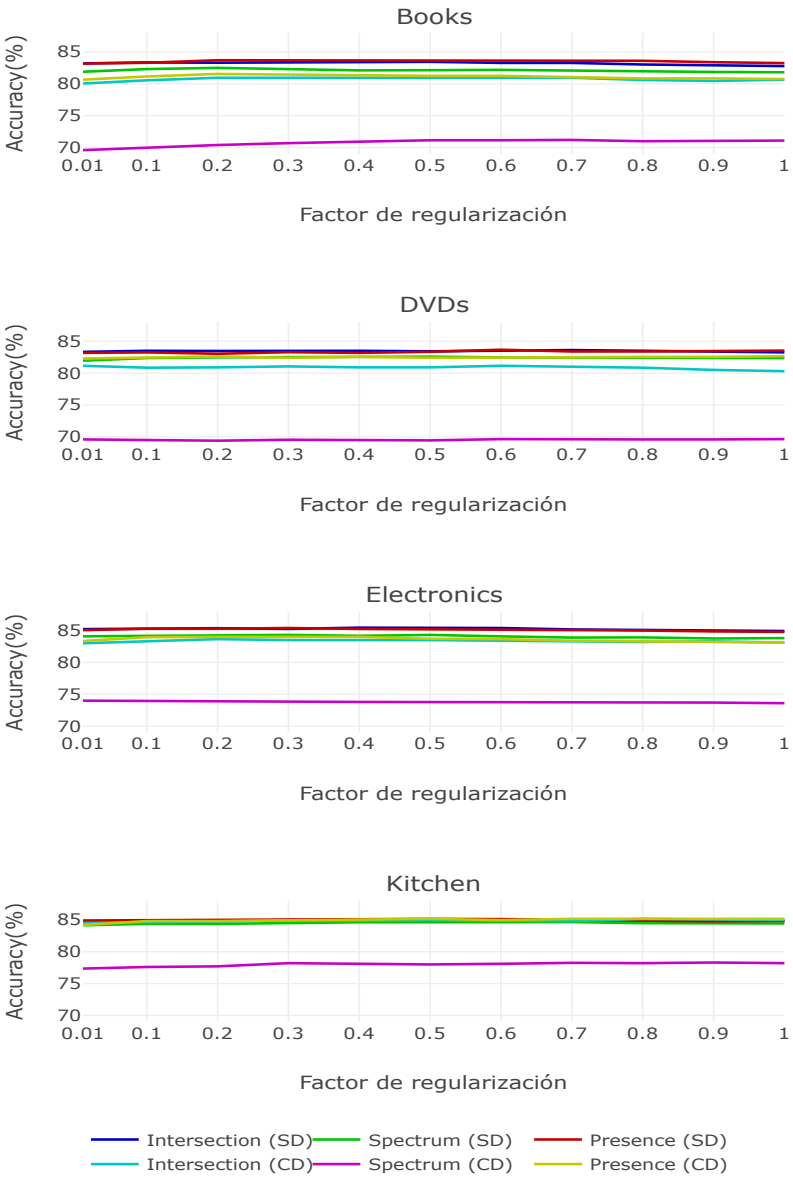

Figura 4.1: Accuracy promedio para todas las particiones dependiendo del kernel y dominio variando el factor de regularización de KDA α.

4.1.4. Clasificación de la polaridad monodominio

Método	**Books**	**DVDs**	**Electronics**	**Kitchen**
KE-Meta	**83.5**	82.3	82.6	84.2
SST	80.4	82.4	84.4	**87.7**
CWL	**82.6**	80.9	**85.9**	85.7
word n-g	80.5	81.7	80.3	81.9
SK($k_p^{0/1}$)	**83.8**	**84.8**	**86.2**	85.5
SK($k_p^{\cap}$)	**83.8**	**84.6**	**86.6**	85.4
SK(k_p)	**82.7**	82.8	84.7	85.3

Cuadro 4.4: Valores de accuracy (en %) para clasificación de la polaridad monodominio.

En la tabla 4.4 podemos ver los resultados que se han obtenido de aplicar la configuración detallada en el apartado 4.1.3 al método propuesto. Dependiendo del dominio, su contribución al estado del arte varía. Como se ha mencionado anteriormente, utilizamos el método *word n-g* como baseline. KE-Meta sobresale en el dominio Books, SST en Kitchen y CWL en Books y Electronics. En [12] se analiza este hecho y lo justifican por la diferencia en la longitud de las reviews y la riqueza del vocabulario entre los diferentes dominios. También subrayan la mayor estabilidad de los resultados proporcionados por su método KE-Meta entre los dominios.

Los resultados proporcionados por nuestro método para los kernels *presence* e *intersection* consiguen superar esta estabilidad. Además, dependiendo del dominio, los resultados son superiores o iguales a los mejores del estado del arte. A excepción del resultado obtenido para el dominio Kitchen con SST, donde la longitud media de las reviews es menor y esto puede penalizar a los otros métodos. Cabe resaltar que no existen diferencias estadísticamente significativas entre los kernels *presence* e *intersection* respeto al test χ^2 y que el kernel *spectrum* consigue resultados peores en todos los casos. A diferencia de los otros dos kernels, *spectrum* asigna un valor

alto incluso cuando sólo uno de los textos tiene una alta frecuencia de un *n*-grama particular. Esto puede producir representaciones similares para textos que quizás no sean tan cercanos a un nivel léxico, lo que conllevaría la penalización del modelo.

4.1.5. Clasificación de la polaridad transdominio

Método	**Books**	**DVDs**	**Electronics**	**Kitchen**
KE-Meta	77.9	**80.4**	78.9	82.5
SST	76.3	78.3	**83.9**	**85.2**
SCL-MI	74.6	76.3	78.9	82.0
word *n*-g	74.4	79.8	77.1	76.9
SK($k_p^{0/1}$)	**82.0**	**81.9**	**83.6**	**85.1**
SK($k_p^{\cap}$)	**80.7**	**80.7**	**83.0**	**85.2**
SK(k_p)	71.2	69.0	73.7	78.0

Cuadro 4.5: Valores de accuracy (en %) para clasificación de la polaridad transdominio.

Para comparar nuestros resultados en el problema de la clasificación de la polaridad transdominio, seguimos el procedimiento empleado en trabajos anteriores [6][12] utilizando una configuración multifuente, es decir, entrenamos con todos los dominios excepto el que vamos utilizar en fase de clasificación. Podemos ver todos los resultados en la tabla 4.5.

Al igual que en el problema anterior, *word n-g* es nuestro baseline. KE-Meta ofrece los mejores resultados para Books y DVDs y SST en Electronics y Kitchen. Puesto que el método SCL-MI fue diseñado para la versión monofuente del problema de la clasificación de la polaridad transdominio, es posible que el uso de múltiples dominios para su fase de entrenamiento sea la razón de que haya obtenido unos resultados más bajos (aunque todavía competentes).

A pesar de no utilizar muestras del dominio objetivo en la fase de entrenamiento (como requieren otros métodos en su fase de adaptación del dominio), los kernels *presence* e *intersection* vuelven a obtener resultados estadísticamente superiores o equivalentes a los proporcionados por el estado del arte. Esto demuestra que los mapeos no lineales aprendidos por el algoritmo KDA consiguen capturar las peculiaridades léxicas que caracterizan la polaridad independientemente del dominio. Nuevamente, destaca la estabilidad de los resultados entre los diferentes dominios en comparación con los otros métodos.

Método	**D→ B**	**E→ B**	**K→ B**	**B→ D**	**E→ D**	**K→ D**
SK($k_p^{0/1}$)	82.0	72.4	72.7	81.4	74.9	73.6
SK($k_p^{\cap}$)	82.1	72.4	72.8	81.3	75.1	72.9
SK(k_p)	81.1	69.9	71.4	80.0	73.5	71.8
	B→ E	**D→ E**	**K→ E**	**B→ K**	**D→ K**	**E→ K**
SK($k_p^{0/1}$)	71.3	74.4	83.9	74.6	75.4	84.9
SK($k_p^{\cap}$)	71.8	74.5	84.4	74.9	75.1	84.9
SK(k_p)	70.7	72.6	83.9	74.2	74.9	84.5

Cuadro 4.6: Valores de accuracy (en %) para clasificación de la polaridad transdominio con un único dominio fuente. La cabecera sigue el formato "dominio_entrenamiento → dominio_test". B, D, E y K hacen referencia a los dominios Books, DVDs, Electronics y Kitchen, respectivamente.

Por otro lado, el kernel *spectrum* consigue unos resultados notablemente peores. Para analizar este hecho, llevamos a cabo un experimento adicional entrenando con un único dominio fuente. Los resultados pueden verse en la tabla 4.6. La comparación entre los resultados empleando una fuente de datos monodominio y otra mutidominio muestra que los kernels *presence* e *intersection* son capaces, en ocasiones, de explotar diferentes características propias del dominio para obtener mejores resultados. Por ejemplo, los kernels *presence* e *intersection* con el dominio Kitchen, y el kernel *presence* con el dominio DVDs. En los casos en los que utilizar varios dominios co-

mo fuente no proporciona mejores resultados, los resultados al menos se acercan a los del dominio más compatible. Para el caso del kernel *spectrum*, se obtienen resultados competentes cuando el dominio fuente es uno de los más compatibles con el objetivo. Sin embargo, la forma de puntuar de este kernel (como se vio en el capítulo 3) hace que su error se incremente cuando utilizamos una configuración multidominio.

4.2. Cross-Lingual Sentiment Dataset

4.2.1. Corpus

En esta segunda fase del experimento, trabajamos con el corpus Cross-Lingual Sentiment dataset (CLS) [29].[1] Al igual que el anterior corpus, está formado por reviews de Amazon en diferentes dominios. Los dominios que se contemplan son tres: Books, DVDs y Music para los idiomas alemán, francés, inglés y japonés. Por cada dominio e idioma, se proporciona una partición de test y otra de training, con 2000 reviews positivas y 2000 negativas (4000 en total por cada para dominio-idioma). El corpus presenta una serie de información en forma de metadatos tales como: la categoría del producto, el número de estrellas, url, id del artículo, titulo del artículo, nombre del autor y localización, fecha, texto de la review... Como en el experimento anterior, utilizaremos únicamente el texto de cada review y su etiqueta de clase.

4.2.2. Metodología

Para evaluar nuestro método sobre este nuevo corpus, nuevamente utilizamos los kernels *presence* ($K_p^{0/1}$), *intersection* ($K_p^{\cap}$) y *spectrum* (K_p). Puesto

[1] https://www.uni-weimar.de/en/media/chairs/computer-science-department/webis/data/corpus-webis-cls-10/

que ya existen una partición de train y otra de test predefinidas, no se llevará a cabo el método de validación cruzada para la obtención de los resultados finales. Hasta donde sabemos, esta será la primera vez que se utilice este corpus para el problema de la clasificación de la polaridad transdominio en su variante monolingüe. Por este motivo, procedemos al cálculo de dos métodos para la comparación de resultados. Por un lado, calcularemos un baseline con un BoW con ponderación tf-idf y utilizaremos como algoritmo de clasificación un SVM con kernel linear. Además, utilizaremos los vectores FastText [5][2] preentrenados de Facebook. Siguiendo el estado del arte, para cada palabra de cada review obtendremos su representación vectorial. Cada review vendrá representada por la media de los vectores de todas las palabras que la forman [24]. Por último, utilizaremos un SVM con kernel lineal como algoritmo de clasificación.

4.2.3. Selección de parámetros

Utilizando nuevamente el kernel *presence* y un parámetro de regulación α de 0.2, ejecutamos las pruebas para diferentes combinaciones de longitudes de n-gramas para los idiomas alemán, francés y japonés. Basándonos en las pruebas anteriores, utilizamos combinaciones incluidas en el rango $[4-9]$ para el estudio. Los resultados para cada idioma y dominio pueden verse en la tabla 4.7. Como puede verse, los resultados para el idioma japonés difieren mucho de los obtenidos por las otras dos lenguas. Esto se debe a que su representación no utiliza el alfabeto románico, sino que hace uso de kanjis y kanas. Estas unidades hacen referencia a palabras completas y sílabas, por lo que cada elemento contiene mucha más información léxica que un carácter del alfabeto románico. Por este motivo, consideramos que es conveniente utilizar un rango de longitud de n-gramas menor que para los otros casos. Así pues, volvemos a repetir los experimentos para esta lengua utilizando combinaciones comprendidas en el rango $[1-6]$. Los resultados obtenidos pueden verse en la tabla 4.8.

[2] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

	DE				**FR**				**JP**			
n-grama	B	D	M	**Media**	B	D	M	**Media**	B	D	M	**Media**
4	81.25	80.70	83.35	81.77	81.85	83.00	83.80	82.88	77.20	77.65	77.80	77.55
4-5	83.20	82.90	83.65	83.25	82.35	84.10	85.40	83.95	76.20	76.85	77.15	76.73
4-6	83.95	83.30	84.25	83.83	82.55	84.05	86.10	84.23	75.70	75.70	76.75	76.05
4-7	83.95	83.65	84.55	84.05	83.10	84.50	86.45	84.68	75.35	75.55	76.55	75.82
4-8	84.25	83.75	84.90	84.30	83.15	84.65	86.50	84.77	74.95	75.10	76.60	75.55
4-9	84.85	83.80	85.10	**84.58**	83.25	84.60	86.55	84.80	74.85	74.95	76.55	75.45
5	84.15	82.85	83.60	83.53	82.75	83.50	85.45	83.90	74.30	74.25	74.70	74.42
5-6	84.30	83.35	84.05	83.90	82.45	83.75	85.75	83.98	73.40	73.25	74.65	73.77
5-7	84.45	83.55	84.50	84.17	82.75	84.20	85.90	84.28	72.80	72.90	74.40	73.37
5-8	84.80	83.65	85.10	**84.52**	82.60	84.85	86.25	84.57	72.55	72.90	74.20	73.22
5-9	85.05	83.75	85.00	**84.60**	83.15	84.70	86.75	**84.87**	72.05	72.70	74.30	73.02
6	84.15	82.85	84.00	83.67	82.10	83.60	85.30	83.67	70.20	70.70	72.60	71.17
6-7	84.85	83.25	84.25	84.12	82.50	84.10	85.80	84.13	69.00	69.80	71.80	70.20
6-8	84.75	83.55	83.90	84.07	82.70	84.80	86.50	84.67	68.70	69.85	71.80	70.12
6-9	84.95	84.00	84.25	84.40	82.90	84.85	87.05	**84.93**	68.25	69.80	71.50	69.85
7	84.20	83.20	83.30	83.57	82.65	84.70	85.70	84.35	66.00	68.65	69.35	68.00
7-8	84.45	83.60	84.05	84.03	81.95	84.90	85.95	84.27	65.15	68.15	67.60	66.97
7-9	84.60	83.40	84.00	84.00	82.40	84.60	86.30	84.43	64.95	67.45	66.55	66.32
8	84.30	83.25	84.00	83.85	81.95	84.65	86.20	84.27	63.10	64.85	64.15	64.03
8-9	84.85	82.80	83.75	83.80	81.60	84.30	86.45	84.12	62.30	64.85	62.35	63.17
9	85.10	82.05	83.40	83.52	81.25	83.60	85.50	83.45	59.75	61.40	59.90	60.35

Cuadro 4.7: Resultados de accuracy (en %) obtenidos con la función kernel presence y un valor α de 0.2 con cada una de las combinaciones de longitudes de n-gramas establecidas, para cada dominio e idioma. B, D y M hacen referencia a los dominios Books, DVDs y Music, respectivamente.

	JP			
n-grama	Books	DVDs	Music	**Media**
1	69.10	72.80	74.65	72.18
1-2	77.95	77.65	79.60	78.40
1-3	79.80	78.85	80.30	79.65
1-4	79.70	79.30	80.85	79.95
2	78.00	77.85	79.35	78.40
2-3	78.75	79.30	80.75	79.60
2-4	79.20	79.55	81.20	79.98
2-5	79.40	79.70	81.40	**80.17**
3	78.90	78.10	79.75	78.92
3-4	78.65	78.55	80.25	79.15
3-5	78.90	78.60	79.60	79.03
1-6	80.10	80.25	81.15	**80.50**
2-6	79.25	79.65	81.35	80.08
3-6	79.05	78.40	79.20	78.88

Cuadro 4.8: Resultados de accuracy (en %) obtenidos con la función kernel presence y un valor α de 0.2 con cada una de las combinaciones de longitudes de n-gramas establecidas, para cada dominio en la lengua japonesa.

Kernel	EN	DE	FR	JP
$k_p^{0/1}$	5-8	5-9	6-9	1-6
$k_p^{\cap}$	5-8	4-9	6-9	1-6
k_p	5-8	5-8	6-9	2-5

Cuadro 4.9: Mejores combinaciones de n-gramas para cada kernel y lenguaje.

Finalmente, para cada lenguaje seleccionamos las mejores combinaciones de longitudes de *n*-gramas (resaltadas en negrita en las tablas) y volvemos a repetir los experimentos únicamente con estas combinaciones para cada kernel de modo que, para cada idioma y kernel, se utilizan unas longitudes de *n*-gramas diferentes. Para el inglés reutilizamos las combinaciones

calculadas en la primera fase de la evaluación. Las mejores combinaciones se muestran en la tabla 4.9.

		Max α Monodominio				**Max α Transdominio**			
		DE	**FR**	**JP**	**EN**	**DE**	**FR**	**JP**	**EN**
$\mathbf{k}_p^{\cap}$	**Books**	0.2	0.4	0.2	0.2	0.1	0.01	0.01	0.7
	DVDs	0.1	0.1	0.01	0.01	0.3	0.4	0.2	0.7
	Music	0.2	0.2	0.1	0.2	0.3	0.2	0.01	0.1
$\mathbf{k}_p$	**Books**	0.1	0.2	0.01	0.1	0.4	0.01	0.01	0.5
	DVDs	0.2	0.1	0.1	0.01	0.2	0.1	0.2	0.3
	Music	0.2	0.01	0.1	0.3	0.4	0.3	0.01	0.3
$\mathbf{k}_p^{0/1}$	**Books**	0.5	0.5	0.2	0.1	0.2	0.01	0.3	0.9
	DVDs	0.01	0.01	0.1	0.3	0.6	0.4	0.6	0.6
	Music	0.1	0.1	0.4	0.2	0.1	0.7	0.01	1

Cuadro 4.10: Valores de α que proporcionan los mejores resultados para cada lenguaje, dominio y kernel.

Teniendo las combinaciones de longitudes de n-gramas ya seleccionadas para cada kernel y lenguaje, se procede al tuneo del parámetro regularizador de KDA α. Nuevamente, se prueban valores comprendidos en el rango [0.01, 1.0]. La variación de resultados para los diferentes valores de alpha pueden verse en las figuras 4.2 4.3 4.4 4.5. Se escoge el mejor valor de α para cada lenguaje, kernel, dominio y variante del problema (monodominio y transdominio). En la tabla 4.10 se muestran los valores seleccionados para cada caso.

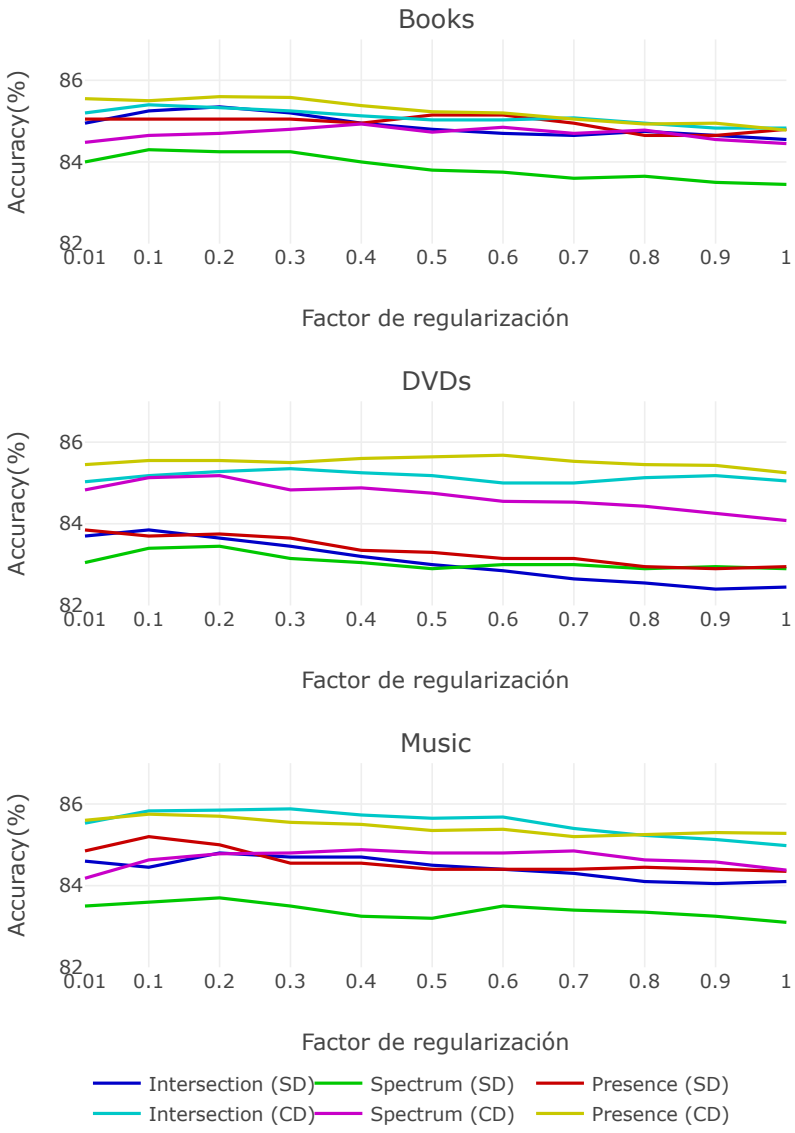

Figura 4.2: Accuracy (en %) para todas las particiones dependiendo del kernel y dominio variando el factor de regularización de KDA α para el idioma alemán.

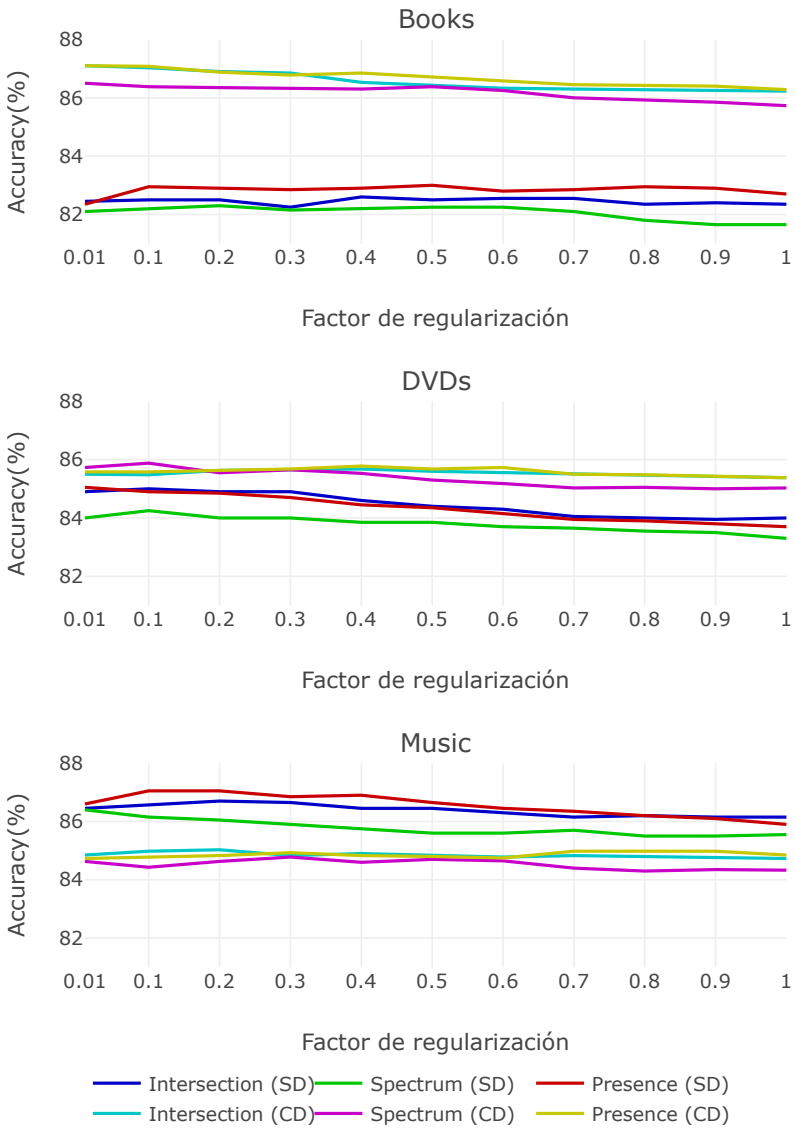

Figura 4.3: Accuracy (en %) para todas las particiones dependiendo del kernel y dominio variando el factor de regularización de KDA α para el idioma francés.

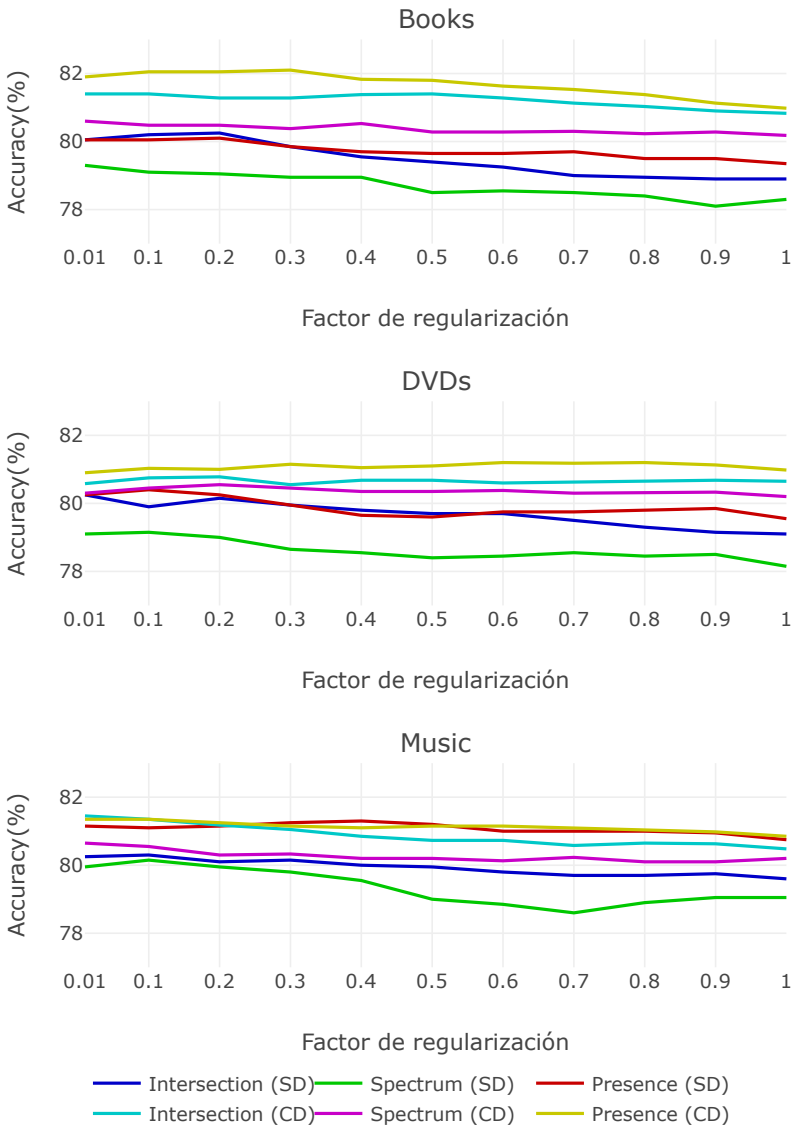

Figura 4.4: Accuracy (en %) para todas las particiones dependiendo del kernel y dominio variando el factor de regularización de KDA α para el idioma japonés.

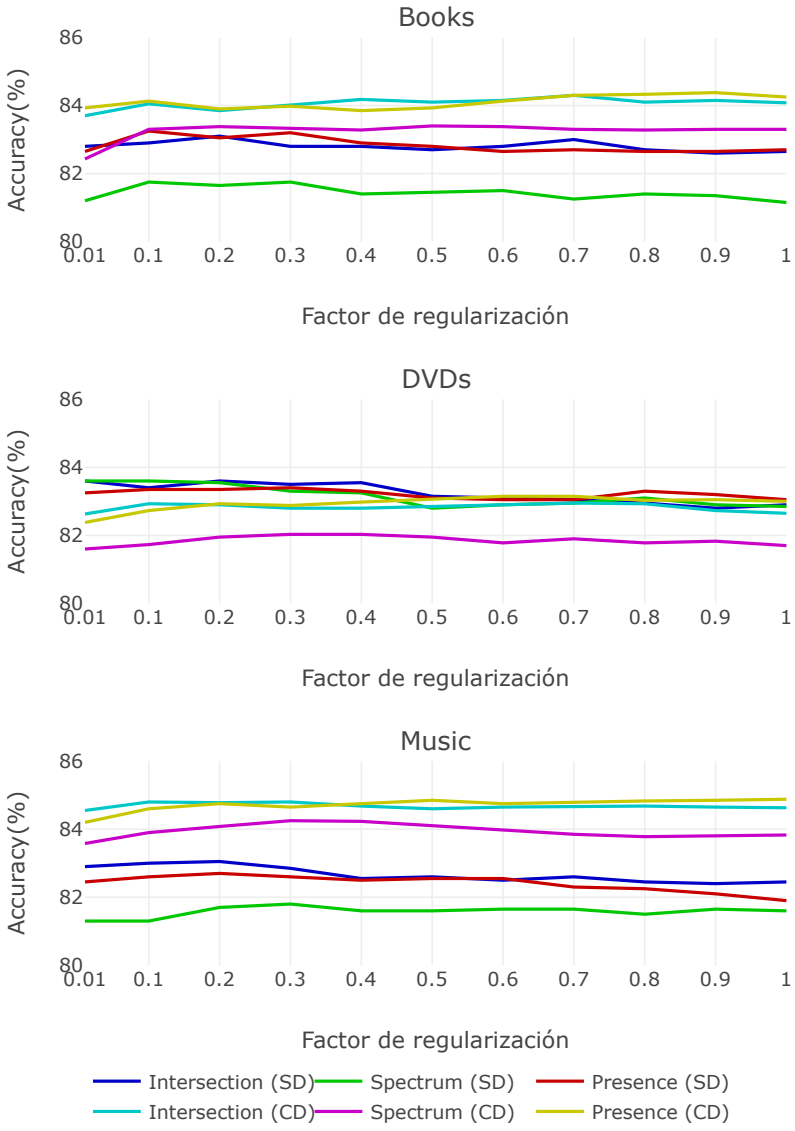

Figura 4.5: Accuracy (en %) para todas las particiones dependiendo del kernel y dominio variando el factor de regularización de KDA α para el idioma inglés.

4.2.4. Clasificación de la polaridad monodominio

Language	EN			DE			FR			JP		
Method	Books	DVDs	Music	Books	DVDs	Music	Books	DVDs	Music	Books	DVDs	Music
tf-idf+SVM	81.4	80.7	81.6	83.2	81.2	81.3	84.1	84.0	85.6	77.4	79.7	79.2
FT+SVM	79.6	77.8	78.8	79.6	79.3	79.2	80.3	79.9	77.8	73.4	73.8	75.5
SK($k_p^{0/1}$)	82.6	82.4	**82.9**	**86.4**	**83.2**	**84.5**	84.6	**85.3**	**86.3**	**80.4**	**81.9**	**81.6**
SK($k_p^{\cap}$)	**82.8**	83.0	82.7	86.3	82.8	84.1	84.3	85.0	86.0	80.1	81.8	80.8
SK(k_p)	82.4	**83.2**	81.8	85.5	81.9	84.0	**84.8**	84.8	86.0	80.2	80.1	80.7

Cuadro 4.11: Resultados para clasificación de la polaridad monodominio utilizando el dataset CLS (en %).

En esta sección evaluamos y comparamos los resultados obtenidos para el problema planteado a nivel monodominio con el CLS corpus. Los resultados obtenidos se muestran en la tabla 4.11. Para el caso anterior, los resultados de los diferentes modelos con los que se comparan los nuestros se han extraído de la literatura. Puesto que no ha sido posible obtener los sistemas por parte de sus autores, para este dataset no es posible la comparación con esos mismos modelos. Como podemos ver, SK consigue mejorar los resultados proporcionados por los otros dos modelos en todos los casos. Este hecho pone de manifiesto el potencial de los SK al emplearlos tanto con textos en alfabeto románico como en alfabeto japonés. Exceptuando el dominio DVDs inglés y Books francés, que consiguen resultados ligeramente superiores con el kernel *spectrum*, y el dominio Books inglés que los obtiene con el kernel *intersection*, en todos los casos se obtienen mejores resultados utilizando el kernel *presence*. De nuevo, como ya se vio en la primera fase de los experimentos, destaca la estabilidad de los resultados entre los dominios para el inglés. Además, vemos que este hecho es extrapolable a los otros lenguajes.

4.2.5. Clasificación de la polaridad transdominio

Language	EN			DE			FR			JP		
Method	Books	DVDs	Music	Books	DVDs	Music	Books	DVDs	Music	Books	DVDs	Music
tf-idf+SVM	78.7	79.3	80.2	80.7	80.6	80.2	**82.4**	83.3	81.4	77.3	77.4	78.7
FT+SVM	80.0	75.7	77.8	79.0	75.4	77.9	78.2	79.5	77.0	71.3	73.4	73.7
SK($k_p^{0/1}$)	81.4	81.5	**81.8**	**84.4**	81.4	**83.6**	82.2	84.4	85.4	**79.9**	80.6	**81.1**
SK($k_p^{\cap}$)	**81.5**	**81.6**	81.5	84.0	**82.5**	82.7	82.1	**84.5**	**85.5**	79.8	80.5	80.9
SK(k_p)	79.9	81.0	81.7	82.6	81.3	82.4	**82.4**	83.6	84.7	79.4	**80.9**	79.2

Cuadro 4.12: Resultado para clasificación de la polaridad transdominio utilizando el dataset CLS (en %).

En este apartado, como se hizo en la primera fase, entrenamos con todos los dominios excepto el que se utiliza como partición de test. Los resultados obtenidos se muestran en la tabla 4.12. Al igual que para los resultados monodominio, SK supera los otros dos modelos comparados. Esto refuerza de nuevo el hecho de que SK tiene un gran potencial cuando se utiliza con idiomas que emplean los alfabetos románico y japonés. En promedio los mejores resultados se obtienen con los kernels *presence* e *intersection*, exceptuando un par de casos donde los resultados con el kernel *spectrum* son ligeramente superiores (dominio Books para francés y dominio DVDs para japonés).

	Books	**DVDs**	**Electronics**	**Kitchen**
EN	175	190	113	96

Cuadro 4.13: Longitudes medias por review para el Multi-Domain Sentiment Dataset

	Books	DVDs	Music
DE	129	133	118
FR	99	100	110
EN	155	162	129
JP	121	119	121

Cuadro 4.14: Longitudes medias por review para el dataset CLS

Aunque en general en este caso los resultados con el kernel *spectrum* también son peores, como sucedió en la versión transdominio de los experimentos realizados con el Multi-Domain Sentiment Daset, no distan tanto de los proporcionados por los otros dos kernels como sucedía en ese caso para los dominios Books y DVDs. Esto puede deberse a las diferencias entre las propiedades de las instancias de cada dataset. En [12] se evaluó la longitud media por review para el Multi-Domain Sentiment Dataset. Los dominios que proporcionaban las reviews más largas obtenían los peores resultados (tabla 4.13). Analizando la longitud para las reviews del dataset CLS (tabla 4.14) vemos que las reviews son, por lo general, más cortas. Lo que puede estar influyendo en estos mejores resultados.

Capítulo 5

Conclusiones

A lo largo de esta tesis se ha tratado de estudiar en profundidad el desempeño del uso de String Kernels en la tarea de clasificación de la polaridad, tanto en su variante monodominio como transdominio. Hemos visto con detenimiento cuáles son los métodos que actualmente conforman el estado del arte para este problema. Con las herramientas y conocimientos adquiridos durante el desarrollo del máster, hemos entrenado nuestros modelos base para obtener resultados de partida con los que comparar los proporcionados por nuestro método propuesto. Se ha explicado y evaluado un nuevo método para tratar de obtener mejores resultados. Se ha dado una explicación más detallada de las funciones String Kernels utilizadas, del método Kernel Discriminant Analysis de transformación y la técnica empleada para clasificar. Hemos analizado el comportamiento de los kernels *presence*, *intersection* y *spectrum* utilizando el algoritmo de reducción de dimensionalidad Kernel Discriminant Analysis.

Los resultados obtenidos para el Multi-Domain Sentiment Dataset se han mostrado y comparado con los proporcionados por los diferentes métodos que conforman el estado del arte. Como se ha podido ver, los kernels *presence* e *intersection* conseguían mejorar estos resultados para las dos variantes del problema. Además, los resultados que se obtenían han mostrado

ser mucho más estables entre los diferentes dominios que los proporcionados por los otros métodos.

También hemos mostrado como el espacio de transformación no lineal del Kernel Discriminant Analysis conseguía capturar las peculiaridades léxicas que caracterizan la polaridad de una forma independiente del dominio. Este hecho ha sido el que ha permitido a nuestro método sobresalir en la comparativa con los métodos analizados en transdominio sin necesitar textos adicionales del dominio objetivo para la fase de entrenamiento. Por otro lado, hemos visto que para los kernels *presence* e *intersection*, utilizar múltiples dominios para entrenamiento en la variante transdominio del problema proporcionaba resultados mejores o similares que simplemente utilizando el dominio más compatible. Queremos remarcar la relevancia que puede tener conseguir buenos resultados de esta forma para la industria, puesto que es más fácil aprender de múltiples dominios independientemente de cuál sea el objetivo que tener que detectarlo y seleccionar en consecuencia. Otra ventaja que aporta el uso de String Kernels es que permite trabajar con millones de características, ya que la función kernel consigue compactar la información permitiendo, de este modo, el uso de tantos n-gramas como sea necesario.

Por otro lado, también se ha tratado de estudiar cuál era su comportamiento al enfrentarse a otros lenguajes. Para ello, se ha utilizado por primera vez en esta tarea el dataset Cross-Lingual Sentiment, el cual nos ha proporcionado textos en alemán, francés, inglés y japonés. Esta última lengua era de especial interés por no compartir el mismo alfabeto con el resto de lenguas europeas. Hemos evaluado la importancia de una correcta selección de longitud de n-gramas para la configuración de nuestro modelo según el lenguaje. Como hemos podido comprobar, las combinaciones de longitudes elegidas para el japonés han sido significativamente inferiores a las seleccionadas para los lenguajes con alfabeto románico. Tanto para el inglés como para los nuevos lenguajes estudiados se han obtenido de nuevo muy buenos resultados, superando los obtenidos por los diferentes baselines. Al igual que para el inglés, los mejores resultados se han obteni-

do en promedio utilizando los kernels *presence* e *intersection*. Nuevamente, la estabilidad de los resultados obtenidos entre los diferentes dominios es remarcable.

Apéndice A

Publicaciones

A continuación mostramos los artículos publicados en congresos dentro del marco de esta tesis:

- Rosa M Giménez-Pérez, Marc Franco-Salvador y Paolo Rosso. "Single and Cross-domain Polarity Classification using String Kernels". En: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Volume 2: Short Papers (EACL 2017)*. Valencia, Spain, 2017, pág. 558 (**CORE A**).
- Rosa M Giménez-Pérez, Marc Franco-Salvador y Paolo Rosso. "String Kernels for Polarity Classification: A Study Across Different Languages". En: *International Conference on Applications of Natural Language to Information Systems (NLDB 2018)*. Springer. 2018, págs. 489-493 (**CORE C**).

Bibliografía

[1] Miguel A Álvarez-Carmona, Marc Franco-Salvador, Esaú Villatoro-Tello, Manuel Montes-y Gómez, Paolo Rosso y Luis Villaseñor-Pineda. "Semantically-informed distance and similarity measures for paraphrase plagiarism identification". En: *Journal of Intelligent & Fuzzy Systems* Preprint (2018), págs. 1-8.

[2] Gaston Baudat y Fatiha Anouar. "Generalized discriminant analysis using a kernel approach". En: *Neural computation* 12.10 (2000), págs. 2385-2404.

[3] John Blitzer, Mark Dredze y Fernando Pereira. "Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification". En: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*. Prague, Czech Republic: Association for Computational Linguistics, 2007, págs. 440-447.

[4] John Blitzer, Ryan McDonald y Fernando Pereira. "Domain adaptation with structural correspondence learning". En: *Proceedings of the 2006 conference on empirical methods in natural language processing (EMNLP 2006)*. Association for Computational Linguistics. 2006, págs. 120-128.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin y Tomas Mikolov. "Enriching word vectors with subword information". En: *arXiv preprint arXiv:1607.04606* (2016).

[6] Danushka Bollegala, David Weir y John Carroll. "Cross-domain sentiment classification using a sentiment sensitive thesaurus". En: *IEEE transactions on knowledge and data engineering* 25.8 (2013), págs. 1719-1731.

[7] Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer y Paul S Roossin. "A statistical approach to machine translation". En: *Computational linguistics* 16.2 (1990), págs. 79-85.

[8] Deng Cai, Xiaofei He y Jiawei Han. "Speed up kernel discriminant analysis". En: *The VLDB Journal—The International Journal on Very Large Data Bases* 20.1 (2011), págs. 21-33.

[9] Corinna Cortes y Vladimir Vapnik. "Support-vector networks". En: *Machine learning* 20.3 (1995), págs. 273-297.

[10] Cedric De Boom, Steven Van Canneyt, Thomas Demeester y Bart Dhoedt. "Representation learning for very short texts using weighted word embedding aggregation". En: *Pattern Recognition Letters* 80 (2016), págs. 150-156.

[11] Mark Dredze, Koby Crammer y Fernando Pereira. "Confidence-weighted linear classification". En: *Proceedings of the 25th international conference on Machine learning (ICML 2008)*. ACM. 2008, págs. 264-271.

[12] Marc Franco-Salvador, Fermín L Cruz, José A Troyano y Paolo Rosso. "Cross-domain polarity classification using a knowledge-enhanced meta-classifier". En: *Knowledge-Based Systems* 86 (2015), págs. 46-56.

[13] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[14] Rosa M Giménez-Pérez, Marc Franco-Salvador y Paolo Rosso. "Single and Cross-domain Polarity Classification using String Kernels". En: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Volume 2: Short Papers (EACL 2017)*. Valencia, Spain, 2017, pág. 558.

[15] Rosa M Giménez-Pérez, Marc Franco-Salvador y Paolo Rosso. "String Kernels for Polarity Classification: A Study Across Different Languages". En: *International Conference on Applications of Natural Language to Information Systems (NLDB 2018)*. Springer. 2018, págs. 489-493.

[16] Gene H Golub y Christian Reinsch. "Singular value decomposition and least squares solutions". En: *Numerische mathematik* 14.5 (1970), págs. 403-420.

[17] Radu-Tudor Ionescu, Marius Popescu y Aoife Cahill. "Can characters reveal your native language? A language-independent approach to native language identification." En: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. Doha, Qatar: Association for Computational Linguistics, 2014, págs. 1363-1373.

[18] Thorsten Joachims. "Text categorization with support vector machines: Learning with many relevant features". En: *European conference on machine learning (ECML 1998)*. Springer. 1998, págs. 137-142.

[19] Solomon Kullback y Richard A Leibler. "On information and sufficiency". En: *The annals of mathematical statistics* 22.1 (1951), págs. 79-86.

[20] Omer Levy y Yoav Goldberg. "Neural word embedding as implicit matrix factorization". En: *Advances in neural information processing systems (NIPS 2014)*. 2014, págs. 2177-2185.

[21] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini y Chris Watkins. "Text classification using string kernels". En: *Journal of Machine Learning Research* 2.Feb (2002), págs. 419-444.

[22] C.D. Manning y H. Schütze. *Foundations of Statistical Natural Language Processing*.

[23] Sebastian Mika, Gunnar Ratsch, Jason Weston, Bernhard Scholkopf y Klaus-Robert Mullers. "Fisher discriminant analysis with kernels".

En: *Neural networks for signal processing IX, 1999. Proceedings of the 1999 IEEE signal processing society workshop.* Ieee. 1999, págs. 41-48.

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado y Jeff Dean. "Distributed representations of words and phrases and their compositionality". En: *Advances in neural information processing systems (NIPS 2013)*. 2013, págs. 3111-3119.

[25] Tomas Mikolov, Kai Chen, Greg Corrado y Jeffrey Dean. "Efficient estimation of word representations in vector space". En: *CoRR* (2013).

[26] Roberto Navigli y Simone Paolo Ponzetto. "BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network". En: *Artificial Intelligence* 193 (2012), págs. 217-250.

[27] Bo Pang, Lillian Lee y Shivakumar Vaithyanathan. "Thumbs up? Sentiment Classification using Machine Learning Techniques". En: *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, 2002, págs. 79-86.

[28] Marius Popescu y Cristian Grozea. "Kernel Methods and String Kernels for Authorship Analysis". En: *Online Working Notes/Labs/Workshop Papers of the CLEF 2012 Evaluation Labs (CLEF 2012)*. Rome, Italy, 2012.

[29] Peter Prettenhofer y Benno Stein. "Cross-language text classification using structural correspondence learning". En: *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2010)*. 2010, págs. 1118-1127.

[30] Lawrence R Rabiner y Biing-Hwang Juang. *Fundamentals of speech recognition*. Vol. 14. PTR Prentice Hall Englewood Cliffs, 1993.

[31] Francisco Rangel y Paolo Rosso. "On the impact of emotions on author profiling". En: *Information processing & management* 52.1 (2016), págs. 73-92.

[32] Antonio Reyes y Paolo Rosso. "On the difficulty of automatically detecting irony: beyond a simple case of negation". En: *Knowledge and Information Systems* (2013), págs. 1-20.

[33] Xin Rong. "word2vec parameter learning explained". En: *arXiv preprint arXiv:1411.2738* (2014).

[34] Gerard Salton, Edward A Fox y Harry Wu. "Extended Boolean information retrieval". En: *Communications of the ACM* 26.11 (1983), págs. 1022-1036.

[35] Gerard Salton y Michael J McGill. "Introduction to modern information retrieval". En: (1986).

[36] Gerard Salton, Anita Wong y Chung-Shu Yang. "A vector space model for automatic indexing". En: *Communications of the ACM* 18.11 (1975), págs. 613-620.

[37] Conrad Sanderson y Simon Guenter. "Short text authorship attribution via sequence kernels, Markov chains and author unmasking: An investigation". En: *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*. Association for Computational Linguistics. 2006, págs. 482-491.

[38] Bernhard Schölkopf. "The kernel trick for distances". En: *Advances in neural information processing systems (NIPS 2001)*. 2001, págs. 301-307.

[39] Fabrizio Sebastiani. "Machine learning in automated text categorization". En: *ACM computing surveys (CSUR)* 34.1 (2002), págs. 1-47.

[40] David H Wolpert. "Stacked generalization". En: *Neural networks* 5.2 (1992), págs. 241-259.

Single and Cross-domain Polarity Classification using String Kernels

Rosa M. Giménez-Pérez[1], **Marc Franco-Salvador**[1,2], and **Paolo Rosso**[1]
[1] Universitat Politècnica de València, Valencia, Spain
[2] Symanto Research, Nuremberg, Germany
rogipe2@upv.es, marc.franco@symanto.net, prosso@upv.es

Abstract

The polarity classification task aims at automatically identifying whether a subjective text is positive or negative. When the target domain is different from those where a model was trained, we refer to a cross-domain setting. That setting usually implies the use of a domain adaptation method. In this work, we study the single and cross-domain polarity classification tasks from the string kernels perspective. Contrary to classical domain adaptation methods, which employ texts from both domains to detect pivot features, we do not use the target domain for training. Our approach detects the lexical peculiarities that characterise the text polarity and maps them into a domain independent space by means of kernel discriminant analysis. Experimental results show state-of-the-art performance in single and cross-domain polarity classification.

1 Introduction

The polarity classification task, also known as (binary) polarity or sentiment categorisation, aims at identifying whether a subjective text is positive or negative depending on the overall sentiment detected. Single domain polarity classification (Pang et al., 2002) refers to the standard text classification setting (Sebastiani, 2002). The cross-domain level (Blitzer et al., 2007) refers to classify a different domain from that or those where a model was trained.

These tasks have become especially important for business purposes. The vastness and accessibility of the Internet produced a new generation of event and product reviewers. These reviewers employ channels such as blogs, fora or social media. In consequence, companies are highly interested into identifying reviewers' opinions on, for instance, new products in order to improve marketing campaigns.

Although polarity classification tasks can be tackled with text classification methods, it has been proven to be a more challenging task (Pang et al., 2002): sentiment may be expressed more subtly (Reyes and Rosso, 2013) than categories generally recognised with keywords alone. In addition, the cross-domain variant has the additional difficulty of using a different vocabulary among domains. This problem is usually drawn by means of domain adaptation techniques (Ben-David et al., 2007). Most of these techniques exploit pivot features that allow to map vocabularies among domains.

String kernels are known for their good performance in text classification (Lodhi et al., 2002). Recent works with this representation demonstrated its excellent capacity to capture lexical peculiarities of text (Popescu and Grozea, 2012; Ionescu et al., 2014). In this work we study the single and cross-domain polarity classification tasks from the string kernels perspective. The research questions we aim to answer are:

- *What is the performance of string kernels for single and cross-domain polarity classification?* We are interested in the performance of this representation in these specially challenging classification tasks. Despite the use of string kernels is not new at single-domain level (Bespalov et al., 2011), this is, to the best of our knowledge, the first attempt to use them at cross-domain level. This leads us to our next research question.

- *Can this representation classify at cross-domain level without learning from texts of the target domain?* We employ Kernel Discriminant Analysis (Mika et al., 1999) for the classification, which is based on a non-linear space transformation. We aim to clarify if

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 558–563,
Valencia, Spain, April 3-7, 2017.

the lexical peculiarities captured by this approach characterise the polarity of the texts independently of the domain.

In order to answer these questions, we compare our approach with several state-of-the-art methods with the well-known Multi-Domain Sentiment Dataset (Blitzer et al., 2007). Experimental results show state-of-the-art performance in single and cross-domain polarity classification. In addition, the stability of the proposed approach is remarkable among the different evaluated domains.

2 Related Work

In this section we review the state-of-the-art methods which have been evaluated in the Multi-Domain Sentiment dataset. Focused on single-domain polarity classification, the Confidence-Weighted Learning (CWL) (Dredze et al., 2008) is based on updating more aggressively the weights of features with higher confidence. The Structural Correspondence Learning with Mutual Information (SCL-MI) (Blitzer et al., 2007) was the first model evaluating the dataset at cross-domain level. The mutual information was used to select pivot features which are subsequently used for measuring co-occurrence with the rest of the features. Chen et al. (2012) addressed this task, considering the scalability and the computational cost of the approach, with marginalized stacked denoising autoencoders. The use of neural networks has also been proven to be useful for cross-domain classification tasks where unlabeled data from the test domain is employed to extract domain independent features (Ganin et al., 2016). Some approaches have proven to excel both at single and cross-domain levels. Bollegala et al. (2013) proposed the Sentiment-Sensitive Thesaurus (SST) model that groups together words expressing the same sentiment. Recently, the Knowledge-Enhanced Meta classifier (KE-Meta) (Franco-Salvador et al., 2015) combined surface and word sense disambiguation features derived from a semantic network.

3 String Kernels

String Kernels (SK) are functions that measure the similarity of string pairs at lexical level. Their dual representation allows to work with a huge number of character n-grams while keeping the feature space reduced.

In this work, we follow the implementation and formulation of Ionescu et al. (2014).[1] A simple measure of the similarity of two strings s,t is the number of shared substrings of length p. The p-grams kernel is estimated as follows:

$$\mathrm{k}_p(s,t) = \sum_{v \in L^p} f(\mathrm{num}_v(s), \mathrm{num}_v(t)), \quad (1)$$

where $\mathrm{num}_v(s)$ is the number of occurrences of string v as a substring of s, p is the length of v, and L is the alphabet used to generate v. The function $f(x,y)$ variates depending on the type of kernel:

1. $f(x,y) = x \cdot y$ in the p-spectrum kernel;
2. $f(x,y) = \mathrm{sgn}(x) \cdot \mathrm{sgn}(y)$ in the p-grams presence bits kernel;[2]
3. $f(x,y) = \min(x,y)$ in the p-grams intersection bits kernel.

As we can see, the values of $f(.)$ are the highest with the spectrum kernel and the lowest with the presence kernel. This gives us an idea about what these kernels capture. The spectrum kernel offers high values even when the texts are only partially related. The intersection kernel employs the n-gram frequency to provide with a precise lexical similarity measure. Finally, the presence kernel captures the lexical *core meaning* of the texts by smoothing the n-gram repetitions.

Our kernels combine different n-gram lengths[3] (see Section 4.2 for details about our parameter selection) and are normalised as follows:

$$\hat{\mathrm{k}}(s,t) = \frac{\mathrm{k}(s,t)}{\sqrt{\mathrm{k}(s,s) \cdot \mathrm{k}(t,t)}} \quad (2)$$

We perform the classification with Kernel Discriminant Analysis (KDA) (Baudat and Anouar, 2000),[4] which returns the eigenvector matrix U. We compute the feature matrices $Y = KU$ and $Y_\mathrm{t} = K_\mathrm{t}U$, where K and K_t are the training and test instance kernels. For each class c, we create the prototype Y_c as the average of all vectors of Y that correspond to the instances of class c.

[1] http://string-kernels.herokuapp.com/

[2] sgn is the sign function.

[3] We combine the n-gram lengths by adding the kernel values obtained for each n.

[4] We use the following KDA implementation: http://www.cad.zju.edu.cn/home/dengcai/Data/DimensionReduction.html

Finally, we classify each test instance by identifying the class of the prototype with the lowest mean squared error between $Y_t(i)$ and Y_c. Key to our cross-domain classification, without learning from texts of the target domain, is the KDA's space transformation. It employs *the kernel trick* (Schölkopf, 2001) and formulates the task as an eigenvalue problem resolution to learn non-linear mappings which transform our features to a new space that captures the most relevant lexical peculiarities for polarity classification.

4 Evaluation

In this section we evaluate and compare our approach in the single and the cross-domain polarity classification tasks.

4.1 Dataset and Tasks Setting

Dataset We employ the Multi-Domain Sentiment Dataset (v. 2.0) (Blitzer et al., 2007).[5] It contains Amazon product reviews of four different domains: Books (B), DVDs (D), Electronics (E) and Kitchen appliances (K). Each review contains information including a rating in a range of 0 to 5 stars. Reviews rated with more than 3 stars were labeled as positive, and those with less than 3 as negative. There are 1,000 positive and 1,000 negative reviews for each domain.

Methodology We evaluate our approach using the presence ($k_p^{0/1}$), intersection ($k_p^{\cap}$), and spectrum (k_p) kernels. We compare with SST and KE-Meta at single and cross-domain levels (see Section 2). In addition, we compare with CWL at single-domain and with SCL-MI at cross-domain level.[6] Finally, we include as a baseline the combination of word unigram, bigram, and trigram features using a support vector machine classifier with linear kernel (henceforth referred to as word n-g). We perform our evaluation with a stratified 10-fold cross-validation. We use the accuracy of classification as the evaluation metric. Statistically significant results according to a χ^2 test are highlighted in bold.

4.2 Parameter Selection

We adjusted the kernel n-gram length and the KDA's regularisation factor α with a 80-20% splitting over the nine training folds of each cross-validation iteration. We first set α to its default value (0.2) and explored different combinations of n-gram lengths, for $2 \leq n \leq 10$. The best results where obtained when we combined all the n-grams in $5 \leq n \leq 8$. Using that combination, we tested for $\alpha \in [0.01, 1]$. The results notably differed depending on the task setting, training domain, and kernel (see Figure 1). We use the parameters adjusted in this section for the rest of our evaluation.

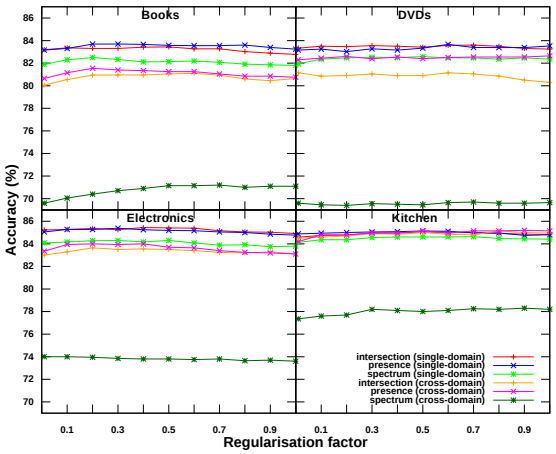

Figure 1: Avg. accuracy among all the fold values depending on the KDA's regularisation factor.

Method	Books	DVDs	Electronics	Kitchen
KE-Meta	**83.5**	82.3	82.6	84.2
SST	80.4	82.4	84.4	**87.7**
CWL	**82.6**	80.9	**85.9**	85.7
word n-g	80.5	81.7	80.3	81.9
SK($k_p^{0/1}$)	**83.8**	**84.8**	**86.2**	85.5
SK($k_p^{\cap}$)	**83.8**	**84.6**	**86.6**	85.4
SK(k_p)	**82.7**	82.8	84.7	85.3

Table 1: Single-domain polarity classification accuracy (in %).

4.3 Single-domain Polarity Classification

In Table 1 we show the single-domain results. As we can see, the state-of-the-art performance differs depending on the domain. The combination of word n-grams makes word n-g the baseline in all the domains. KE-Meta excels with book reviews, SST with kitchen appliance reviews, and CWL with book and electronic reviews. Franco-Salvador et al. (2015) analysed this fact and justified it with the difference in review length and

[5] https://www.cs.jhu.edu/~mdredze/datasets/sentiment/

[6] The results of the compared approaches are taken from Franco-Salvador et al. (2015).

Method	Books	DVDs	Electronics	Kitchen
KE-Meta	77.9	**80.4**	78.9	82.5
SST	76.3	78.3	**83.9**	**85.2**
SCL-MI	74.6	76.3	78.9	82.0
word n-g	74.4	79.8	77.1	76.9
SK($k_p^{0/1}$)	**82.0**	**81.9**	**83.6**	**85.1**
SK($k_p^{\cap}$)	**80.7**	**80.7**	**83.0**	**85.2**
SK(k_p)	71.2	69.0	73.7	78.0

Table 2: Multi-source cross-domain polarity classification accuracy (in %).

vocabulary richness among the evaluated domains. In addition, they highlighted the KE-Meta stability among domains, i.e., their higher lower-bound in accuracy. However, the results of our presence and intersection string kernels are more stable. What is more, depending on the domain, their results are statistically superior or equal to the best obtained by the state of the art. The exception is SST, which obtains the best results in the kitchen domain, where the shorter average review length could penalise other methods. We note that there are not statistically significant differences between the presence and intersection kernels. However, the spectrum kernel obtains lower results in all the cases. In contrast to the other two kernels, the spectrum one assigns a high score even when only one of the texts has a high frequency for a particular n-gram (see Section 3). This produces similar kernel representations for texts which may be not so close at lexical level and, consequently, penalises the model precision.

4.4 Cross-domain Polarity Classification

Following recent works in cross-domain polarity classification (Bollegala et al., 2013; Franco-Salvador et al., 2015), in Table 2 we compare with the state of the art using a multi-source cross-domain setting, i.e., we train with all the domains but the one we classify. Similarly to the single-domain results, word n-g is the baseline, KE-Meta offers higher results in book and DVD reviews, and SST in electronic and kitchen appliance reviews. We note that SCL-MI was designed for single-source cross-domain classification (Blitzer et al., 2007). Therefore, the use of multiple training domains may be the reason of its lower, but still competitive, performance.

Interestingly, despite not using target domain texts for training, the presence and intersection kernels obtain statistically superior or equal results to the best ones obtained by the state of the art. This proves that the non-linear mappings learned by KDA capture the lexical peculiarities that characterise polarity in a domain-independent way. We note again the stability of the results of these kernels and the non-existent statistically significant difference between them. In contrast, the spectrum kernel obtains the lowest results of the table. In order to analyse this fact, we perform an additional experiment where we use a single-source setting to train our cross-domain classifiers. We can see the results in Table 3.

The comparison of the multi-source and the single-source results shows that the presence and intersection kernels are occasionally able to exploit different domain characteristics to obtain better results, e.g. the presence and intersection kernels with kitchen reviews, and the presence kernel with DVDs reviews. Even in cases when the combination of domains do not lead to better results, the results remain close to those of the most compatible training domain; specially with the presence kernel. We note the relevance of the multi-source setting for the industry: it is easier to use multiple domains to learn a domain-independent classifier than to detect each time which is the most appropriated training domain. Finally, we observe that the spectrum kernel has competitive results when the most compatible domain is used for training. However, the aforementioned score characteristics of that kernel (see Sections 3 and 4.3) exponentially increase its error in the multi-source setting.

5 Conclusions

In this paper we studied the single and the cross-domain polarity classification tasks from the string kernels perspective. We analysed the performance of the presence, intersection, and spectrum kernels when classifying with kernel discriminant analysis. Experimental results compared to several state-of-the-art approaches in the Multi-Domain Sentiment Dataset showed state-of-the-art performance for the presence and intersection kernels in both tasks. In addition, these two kernels provided with the most stable results among domains. What is more, we showed that the non-linear space transformations of kernel discriminant analysis captured the lexical peculiarities that characterise polarity in a domain-independent way. This fact

Method	D→B	E→B	K→B	B→D	E→D	K→D
SK($k_p^{0/1}$)	82.0	72.4	72.7	81.4	74.9	73.6
SK($k_p^{\cap}$)	82.1	72.4	72.8	81.3	75.1	72.9
SK(k_p)	81.1	69.9	71.4	80.0	73.5	71.8
	B→E	**D→E**	**K→E**	**B→K**	**D→K**	**E→K**
SK($k_p^{0/1}$)	71.3	74.4	83.9	74.6	75.4	84.9
SK($k_p^{\cap}$)	71.8	74.5	84.4	74.9	75.1	84.9
SK(k_p)	70.7	72.6	83.9	74.2	74.9	84.5

Table 3: SK single-source cross-domain polarity classification accuracy (in %), where each column header follows the "training domain → test domain" format.

allowed our approaches to excel at cross-domain level without learning from texts of the target domain. Finally, the analysis of the single-source and the multi-source cross-domain results proved that the presence kernel tolerates better the inclusion of new training domains in the multi-source cross-domain setting. This fact makes it the recommended option for cross-domain polarity classification.

Future work will investigate further how to employ string kernels for single and cross-domain classification tasks.

Acknowledgments

We thank Ionescu et al. (2014) for their support and comments. The work of the third author was partially supported by the SomEMBED TIN2015-71147-C2-1-P MINECO research project and by the Generalitat Valenciana under the grant ALMAMATER (PrometeoII/2014/030).

References

Gaston Baudat and Fatiha Anouar. 2000. Generalized discriminant analysis using a kernel approach. *Neural computation*, 12(10):2385–2404.

Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. 2007. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137.

Dmitriy Bespalov, Bing Bai, Yanjun Qi, and Ali Shokoufandeh. 2011. Sentiment classification based on supervised latent n-gram analysis. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM'11)*, pages 375–382, Glasgow, Scotland, UK. ACM.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic, June. Association for Computational Linguistics.

Danushka Bollegala, David Weir, and John Carroll. 2013. Cross-domain sentiment classification using a sentiment sensitive thesaurus. *IEEE transactions on knowledge and data engineering*, 25(8):1719–1731.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning (ICML'12)*, pages 767–774, Edinburgh, Scotland.

Mark Dredze, Koby Crammer, and Fernando Pereira. 2008. Confidence-weighted linear classification. In *Proceedings of the 25th International Conference on Machine Learning (ICML'08)*, pages 264–271, Helsinki, Finland. ACM.

Marc Franco-Salvador, Fermín L. Cruz, José A. Troyano, and Paolo Rosso. 2015. Cross-domain polarity classification using a knowledge-enhanced meta-classifier. *Knowledge-Based Systems*, 86:46 – 56.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35.

Radu-Tudor Ionescu, Marius Popescu, and Aoife Cahill. 2014. Can characters reveal your native language? A language-independent approach to native language identification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1363–1373, Doha, Qatar, October. Association for Computational Linguistics.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444.

Sebastian Mika, Gunnar Rätsch, Jason Weston, Bernhard Schölkopf, and Klaus-Robert Müller. 1999. Fisher discriminant analysis with kernels. In *Proceedings of IEEE Neural Networks for Signal Processing Workshop (NNSP'99)*, pages 41–48, Madison, Wisconsin, USA.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 79–86, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Marius Popescu and Cristian Grozea. 2012. Kernel methods and string kernels for authorship analysis. In *Online Working Notes/Labs/Workshop Papers of the CLEF 2012 Evaluation Labs (CLEF'12)*, Rome, Italy.

Antonio Reyes and Paolo Rosso. 2013. On the difficulty of automatically detecting irony: beyond a simple case of negation. *Knowledge and Information Systems*, pages 1–20.

Bernhard Schölkopf. 2001. The kernel trick for distances. *Advances in neural information processing systems*, 13:301–307.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

String Kernels for Polarity Classification: A Study across Different Languages

Rosa M. Giménez-Pérez[1], Marc Franco-Salvador[2], and Paolo Rosso[1]
rogipe2@upv.es, marc.franco@symanto.net, prosso@upv.es

[1] Universitat Politècnica de València, Spain
[2] Symanto Research, Nuremberg, Germany

Abstract. The polarity classification task has as objective to automatically deciding whether a subjective text is positive or negative. Using a cross-domain setting implies the use of different domains for the training and testing. Recently, string kernels, a method which does not employ domain adaptation techniques has been proposed. In this work, we analyse the performance of this method across four different languages: English, German, French and Japanese. Experimental results show the strong potential of this approach independently from the language.

Keywords: string kernels, sentiment analysis, single-domain, cross-domain

1 Introduction

Since the Web 2.0 emergence, the Internet has been providing different channels where people can express their opinions about many products and services. As a result, blogs, fora and social media have become an important information source for companies. As a consequence, there is a high interest in identifying opinions and reviews in order to improve the business they offer.

The task of deciding if those texts are positive or negative depending on the overall sentiment detected is known as sentiment or polarity classification task. Single-domain polarity classification (SD)[6] refers to the standard text classification setting. Cross-domain polarity classification (CD)[2] refers to testing on a different domain from that or those used for training the model. Although this task can be tackled as a common text classification problem, sentiment may be expressed in a more subtle manner. Furthermore, in the CD variant it exists the additional difficulty of using different vocabularies among the domains. For these reasons, solutions based only on bag-of-words representations are not enough.

In [4] the authors studied the performance of string kernels at SD and CD level for English texts with very promising results and showed their capability to capture the lexical peculiarities that characterise the polarity in a domain-independent way.

In order to further investigate string kernels performance in polarity classification, in this work we study their application for other languages, i.e., English, German, French and Japanese. This study includes the analysis of the impact of key parameters such as the string length depending on the alphabet employed.

This is, to the best of our knowledge, the first time that this dataset has been used in the mono-lingual polarity classification task.

2 String Kernels

String Kernels (SK) are functions that measure the similarity of string pairs at lexical level. Their dual representation allows to keep the feature space reduced, even working with a huge number of characteristics. Following the implementation and formulation of Ionescu et al. (2014) [5],[3] the p-grams kernel function counts how many substrings of length p have two strings s and t in common: $\mathrm{k}_p(s,t) = \sum_{v \in L^p} f(\mathrm{num}_v(s), \mathrm{num}_v(t))$, where $\mathrm{num}_v(s)$ is the number of occurrences of string v as a substring of s, p is the length of v, and L is the alphabet used to generate v. The function $f(x,y)$ varies depending on the type of kernel: $f(x,y) = x \cdot y$ in the p-spectrum kernel; $f(x,y) = \mathrm{sgn}(x) \cdot \mathrm{sgn}(y)$ in the p-grams presence bits kernel;$f(x,y) = \min(x,y)$ in the p-grams intersection bits kernel.

Taking into account that the spectrum kernel provides the highest values and presence kernel the lowest, this can gives us an idea about what these kernels capture. The spectrum kernel offers high values even when the texts are only partially related. The intersection kernel employs the n-gram frequency to provide with a precise lexical similarity measure. Finally, the presence kernel captures the lexical *core meaning* of the texts by smoothing the n-gram repetitions.

The kernels we implemented combine different n-gram lengths by adding the kernel values obtained for each n (see Section 3.2 for details about our parameter selection) and are normalised as follows: $\hat{\mathrm{k}}(s,t) = \mathrm{k}(s,t)/\sqrt{\mathrm{k}(s,s) \cdot \mathrm{k}(t,t)}$.

We perform the classification with Kernel Discriminant Analysis (KDA) [1], We compute the feature matrices $Y = KU$ and $Y_\mathrm{t} = K_\mathrm{t}U$, where K and K_t are the training and test instance kernels. For each class c, we create the prototype Y_c as the average of all vectors of Y that correspond to the instances of class c. Finally, we classify each test instance by identifying the class of the prototype with the lowest mean squared error between $Y_\mathrm{t}(i)$ and Y_c.

3 Evaluation

3.1 Dataset and Tasks Setting

We employ the CLS dataset[4] which is formed by Amazon product reviews of three domains (Books, DVDs and Music) in four languages: English, German, French and Japanese. Each language-domain partition is formed by a training and a test set with 1,000 positive and 1,000 negative reviews each one.

To evaluate our approach, we use the presence ($\mathrm{k}_p^{0/1}$), intersection ($\mathrm{k}_p^{\cap}$), and spectrum (k_p) kernels. In order to compare the results, we calculate a baseline using the Tf-idf weighting and Support Vector Machines (SVM) using a linear kernel. In addition, we implement a model based on distributed representations of words. We use the recent Facebook's pretrained FastText [3] vectors.[5] We

[3] http://string-kernels.herokuapp.com/

[4] https://www.uni-weimar.de/de/medien/professuren/medieninformatik/webis/data/webis-cls-10/

[5] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

average the word vectors to represent the instances. We classify using SVM with a linear kernel. In this work, the best results of the tables are highlighted in bold.

3.2 String Kernel Parameter Selection

The kernel n-gram length and the KDA's regularisation factor α are adjusted with a 10-cross-validation over the train partition of each domain. First, we set α to a default value (0.2) and select the presence kernel to analyse the results for different combinations of n-gram lengths. Following the procedure of Giménez-Pérez et al. (2017) [4] we tried combinations for $4 \leq n \leq 9$ for all the evaluated languages. However, during the prototyping, we realised that n-grams within that range are not adequate for languages such as Japanese. The lexical and semantic information included inside a symbol of its alphabet is notably larger than the information included in the Roman alphabet. Therefore, we modified the search space of the Japanese string length to $1 \leq n \leq 6$. Once that parameter was adjusted, we tested different α values between 0.01 and 1.0 and selected the best for every language, domain and kernel in each task.

3.3 Results

First, we evaluate and compare the models at SD level. SK outperform the other two models in all the cases. This manifests their potential for texts written using the Roman and Japanese alphabets. The French Books domain and the English DVDs one work marginally better with the spectrum kernel. The English Books domain shows the best results with the intersection kernel. Excepting those cases, all the other domains and languages obtained the best results using the presence kernel. Giménez-Pérez et al. (2017) [4] proved that this method offers a notable stability among different English domains. That statement is extended here to the rest of languages evaluated.

For CD level, we train with all the domains but the one used in the test partition. Similarly to the single-domain results, SK outperform the two compared models. This reinforces the SK suitability regarding their potential with the Roman and Japanese alphabets. Although the best results are obtained on average with the presence and intersection kernel, the spectrum kernel also obtains competitive results, being even the best option in some cases (French Books and Japanese DVDs domains). Results are shown in Table 1.

4 Conclusions

In this paper we studied the use of string kernels for the single and cross-domain polarity classification task and studied their behaviour across four languages: English, German, French and Japanese. We used for the first time the CLS dataset in mono-lingual polarity classification tasks. We evaluated the intersection, presence and spectrum kernels when classifying with kernel discriminant analysis. We evaluated the importance of the n-gram length selection depending on the language. This showed that the best results for the Japanese alphabet

	Language	EN			GE			FR			JP		
	Method	Books	DVDs	Music	Books	DVDs	Music	Books	DVDs	Music	Books	DVDs	Music
SD	Tf-idf+SVM	81.4	80.7	81.6	83.2	81.2	81.3	84.1	84.0	85.6	77.4	79.7	79.2
	FT+SVM	79.6	77.8	78.8	79.6	79.3	79.2	80.3	79.9	77.8	73.4	73.8	75.5
	SK($k_p^{0/1}$)	82.6	82.4	**82.9**	**86.4**	**83.2**	**84.5**	84.6	**85.3**	**86.3**	**80.4**	**81.9**	**81.6**
	SK($k_p^{\cap}$)	**82.8**	83.0	82.7	86.3	82.8	84.1	84.3	85.0	86.0	80.1	81.8	80.8
	SK(k_p)	82.4	**83.2**	81.8	85.5	81.9	84.0	**84.8**	84.8	86.0	80.2	80.1	80.7
CD	Tf-idf+SVM	78.7	79.3	80.2	80.7	80.6	80.2	**82.4**	83.3	81.4	77.3	77.4	78.7
	FT+SVM	80.0	75.7	77.8	79.0	75.4	77.9	78.2	79.5	77.0	71.3	73.4	73.7
	SK($k_p^{0/1}$)	81.4	81.5	**81.8**	**84.4**	81.4	**83.6**	82.2	84.4	85.4	**79.9**	80.6	**81.1**
	SK($k_p^{\cap}$)	**81.5**	**81.6**	81.5	84.0	**82.5**	82.7	82.1	**84.5**	**85.5**	79.8	80.5	80.9
	SK(k_p)	79.9	81.0	81.7	82.6	81.3	82.4	**82.4**	83.6	84.7	79.4	**80.9**	79.2

Table 1. SD and CD polarity classification accuracy (in %).

are obtained when selecting smaller lengths than the ones employed with the Roman alphabet. The best classification results were obtained on average using the presence and intersection kernel. In addition, the stability of the results among the different evaluated domains was notably high for all the evaluated languages. Finally, string kernels showed strong potential, in all the evaluated languages, at capturing the lexical peculiarities that characterise polarity in a domain-independent way.

Acknowledgements

The work of the third author was partially funded by the Spanish MINECO under the research project SomEMBED (TIN2015-71147-C2-1-P).

References

1. Baudat, G., Anouar, F.: Generalized discriminant analysis using a kernel approach. Neural computation 12(10), 2385–2404 (2000)
2. Blitzer, J., Dredze, M., Pereira, F.: Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In: Proceedings of the Annual Meeting of the Association of Computational Linguistics. pp. 440–447 (2007)
3. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics 5 (2017)
4. Giménez-Pérez, R.M., Franco-Salvador, M., Rosso, P.: Single and cross-domain polarity classification using string kernels. In: Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics. p. 558 (2017)
5. Ionescu, R.T., Popescu, M., Cahill, A.: Can characters reveal your native language? A language-independent approach to native language identification. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 1363–1373 (2014)
6. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? sentiment classification using machine learning techniques. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 79–86 (2002)