Curso: Minería de Textos (2019)
Facultad de Informática
Universidad Nacional de La Plata

Cuestionario

Nota aclaratoria: Este cuestionario consta de 10 ejercicios y un ejerecicio adicional opcional. En los 10 primeros ejercicios se debería responder al menos un 70 % de las preguntas. El ejercicio número 11 se deja como complementario y se recomienda su realización si considera que con los 10 primeros ejercicios no cumplimenta el 70 % requerido. Usted podrá observar un número considerable de hojas en este cuestionario. No se preocupe, la mayoría de estas hojas están dedicadas a ejemplos, explicaciones y material complementario para ayudar en el entendimiento y resolución de los ejercicios. La mayoría de los conceptos involucrados en las preguntas fueron explicados y ejemplificados en clase. No obstante eso, nuevos conceptos importantes también han sido introducidos en este cuestionario con sus respectivas explicaciones, a los fines de complementar el material que se vio durante el dictado del curso. En cualquier caso, si con el material presentado en clase, el sugerido como complementario en este cuestionario, o la información que se hubiera podido obtener por Internet no se logra tener una comprensión adecuada de un tema, no dude en consultarme a cualquiera de los e-mails provistos en la clase 0, y le daré pistas o material adicional para solucionar cualquier duda. Por último, las respuestas a este cuestionario no deberían ser enviadas a esos mismos e-mails más allá del 15 de Noviembre del corriente año. La bibliografía mencionada en este documento está disponible en el archivo `referencias.zip` en el repositorio de Github donde se ha dejado el resto del material del curso.

Parte A: Pre-procesamiento de textos

Ejercicio 1 Tokenización

Introducción

El proceso de *tokenization* usualmente particiona el texto en "tokens" (palabras, números, etc) que lucen razonables como unidades básicas de procesamiento de los documentos. A modo de ejemplo, a continuación

vemos como ésto puede ser logrado en `scikit-learn` con un ejemplo sencillo de particionado en palabras de dos cadenas (strings) de caracteres:

```
In [1]: from sklearn.feature_extraction.text import CountVectorizer
        frase = ["The fool man thinks he is wise,",
                 "but the wise man knows himself to be a fool"]
        print("frase:\n{}".format(frase))

        cv = CountVectorizer(ngram_range=(1, 1)).fit(frase)
        print("Tamaño de vocabulario: {}".format(len(cv.vocabulary_)))
        print("Vocabulario:\n{}".format(cv.get_feature_names()))
```

```
frase:
['The fool man thinks he is wise,', 'but the wise man knows himself to be a fool']

Tamaño de vocabulario: 12
Vocabulario:
['be', 'but', 'fool', 'he', 'himself', 'is', 'knows', 'man', 'the', 'thinks', 'to', 'wise']
```

Sin embargo, la tokenización puede generar como "tokens" a n-gramas (secuencias contiguas) de palabras o de caracteres. A modo de ejemplo, veamos cómo las cadenas de caracteres del ejemplo previo podrían ser divididas como 2-gramas (*bigramas*) de *palabras* y 3-gramas (*trigramas*) de caracteres:

```
In [2]: cv = CountVectorizer(ngram_range=(2, 2)).fit(frase)
        print("Tamaño de vocabulario: {}".format(len(cv.vocabulary_)))
        print("Vocabulario:\n{}".format(cv.get_feature_names()))
```

```
Tamaño de vocabulario: 14
Vocabulario:
['be fool', 'but the', 'fool man', 'he is', 'himself to', 'is wise', 'knows himself', 'man knows',
'man thinks', 'the fool', 'the wise', 'thinks he', 'to be', 'wise man']
```

```
In [3]: cv = CountVectorizer(analyzer='char',ngram_range=(3, 3)).fit(frase)
        print("Tamaño de vocabulario: {}".format(len(cv.vocabulary_)))
        print("Vocabulario:\n{}".format(cv.get_feature_names()))
```

```
Tamaño de vocabulario: 56
Vocabulario:
[' a ', ' be', ' fo', ' he', ' hi', ' is', ' kn', ' ma', ' th', ' to', ' wi', 'a f', 'an ', 'be ',
'but', 'e a', 'e f', 'e i', 'e m', 'e w', 'elf', 'f t', 'foo', 'he ', 'him', 'hin', 'ims', 'ink',
'is ', 'ise', 'kno', 'ks ', 'l m', 'lf ', 'man', 'mse', 'n k', 'n t', 'nks', 'now', 'o b', 'ol ',
'ool', 'ows', 's h', 's w', 'se ', 'se,', 'sel', 't t', 'the', 'thi', 'to ', 'ut ', 'wis', 'ws ']
```

Desarrollo

1. Suponga que se tienen las siguientes cadenas en español:

   ```
   "su auto grande pasó velozmente"
   "tu auto rojo es más caro"
   ```

   Dé el vocabulario (y su tamaño) que se generaría cuando la tokenización se realiza por palabra (1-grama), bigrama de palabras y 5-gramas de caracteres.

2. Dé su opinión de cuales serían las limitaciones de la primera forma de particionado (1-grama) y los posibles patrones que permitiría capturar los restantes dos enfoques

3. El uso de 3-gramas de caracteres es un enfoque que ha dado buenos resultados en el idioma inglés. ¿Considera usted que en el español podría ser mejor utilizar un número n distinto de caracteres contiguos? Justifique.

Ejercicio 2 Normalización (truncado y lematización) y POS-tagging

1. Utilizando alguno de los "stemmers on-line" provistos en el sitio `http://textanalysisonline.com/` (por ejemplo `http://textanalysisonline.com/nltk-porter-stemmer`), chequee cual sería el resultado que el stemmer de Porter arrojaría para la cadena:

   ```
   "Some housewives felt scared when they heard a lot of mice"
   ```

2. Describa cual sería el resultado que a su criterio produciría un sistema de lematización con la misma cadena.

3. Describa y explique cual es el resultado que el POS-tagger online de *spaCy* (disponible en `http://textanalysisonline.com/spacy-pos-tagging`) produce con la misma cadena.

Ejercicio 3 Sentidos de las palabras y Wordnet

1. Defina con sus palabras a qué se denomina *synset* en Wordnet, y en que consisten las relaciones semánticas de *hiperonimia*, *hiponimia* y *meronimia*.

2. Descargando y utilizando el sistema Wordnet 2.1 disponible en `https://wordnet.princeton.edu/download`, determine a que distancia (en número de conceptos recorridos) se encuentra el primer

sentido del sustantivo (noun) nose del concepto organ siguiendo la relación de *hiperonimia* ("es una clase de") y a que distancia del concepto de body siguiendo la relación de *holonimia* ("es una parte de"). A modo de ejemplo: si buscamos la palabra cat (gato) y usamos el botón "Noun" para ver las distintas relaciones semánticas como sustantivo, veremos que siguiendo la relación de hiperonimia, "cat" se encuentra a una distancia de 5 (5 saltos) en la relación de hiperonimia, dada por la siguiente secuencia:

`cat` ⇒ `feline` ⇒ `carnivore` ⇒ `placental` ⇒ `mammal` ⇒ `vertebrate`

Ejercicio 4 - Reconocimiento de Entidades nombradas

Utlizando el reconocedor de entidades nombradas de spaCy disponible en `http://textanalysisonline.com/spacy-named-entity-recognition-ner`, describa cual es el resultado del mismo al aplicarlo a las siguientes sentencias:

```
"Washinton was born into slavery on the farm of James Burroughs."
```

```
"In June, Washington passed a primary seatbelt law."
```

Explique además el porqué del etiquetado diferente de `Washington` en los dos casos.

Parte B: Representación de documentos

Ejercicio 5 (Representación Bolsa de Palabras (BoW))

Introducción

Suponga que se tienen los siguientes 5 documentos, representados como 5 strings:

```
In [4]: documentos = ["Los jugadores llevaban la pelota oficial al estadio",
                      "Durante el partido, los jugadores no tocaron la pelota",
                      "El estadio de Wembley es el estadio más conocido de Londres",
                      "Los senadores pudieron votar la ley en el congreso",
                      "Todos los senadores del partido opositor pidieron vetar la ley"]
```

Asuma además que estos documentos serán vectorizados, de manera tal que se tokenizará por palabra y además se eliminarán algunas palabras de paro en español, de la siguiente manera:

```
In [5]: sw_espa = ["el", "la", "los", "las", "al", "de", "en", "del"]

        cv = CountVectorizer(ngram_range=(1, 1),stop_words=sw_espa).fit(documentos)
        print("Tamaño de vocabulario: {}".format(len(cv.vocabulary_)))
        print("Vocabulario:\n{}".format(cv.get_feature_names()))
```

```
Tamaño de vocabulario: 23
Vocabulario:
['congreso', 'conocido', 'durante', 'es', 'estadio', 'jugadores', 'ley', 'llevaban', 'londres', 'más',
'no', 'oficial', 'opositor', 'partido', 'pelota', 'pidieron', 'pudieron', 'senadores', 'tocaron',
'todos', 'vetar', 'votar', 'wembley']
```

Como se puede observar, el tamaño del vocabulario para esos 5 documentos (sin las palabras de paro especificadas) es de 23 palabras.

Si convertimos estos documentos a represesentación BoW (vectorial), la matriz resultante se muestra a continuación:

```
In [6]: print("Documentos: {}".format(repr(documentos)))
        print()
        bag_of_words = cv.transform(documentos)
        print("Representación vectorial de los documentos (TF):\n{}".format(bag_of_words.toarray()))
```

```
Documentos: ['Los jugadores llevaban la pelota oficial al estadio', 'Durante el partido, los jugadores no
tocaron la pelota', 'El estadio de Wembley es el estadio más conocido de Londres', 'Los senadores pudieron
votar la ley en el congreso', 'Todos los senadores del partido opositor pidieron vetar la ley']

Representación vectorial de los documentos (TF):
[[0 0 0 0 1 1 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 1 0 0 1 0 0 0 0 1 0 0 1 1 0 0 0 1 0 0 0 0]
 [0 1 0 1 2 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1]
 [1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1 0 1 0 1 1 1 0 0]]
```

Como se puede observar, la dimensión de esta matriz está dada por el número de documentos (5) $\times$ el número de términos en el vocabulario (23). Además, se observa que el pesado de los téminos es uno de los más sencillos, ya que sólo se registra la frecuencia del término en el documento (pesado TF). Así, por ejemplo, vemos que en el tercer documento, el quinto término (`estadio`) ocurre dos veces. También es directo observar que este tipo de representación tiende a generar matrices "ralas" (dispersas) ya que la mayoría de los elementos que representan un documento son 0's. Esto es mucho más evidente aún en colecciones donde el tamaño del vocabulario supera los 100.000 términos, y cada documento tiene unas pocas decenas o centenas de palabras.

Desarrollo

1. Especifique cómo hubiera sido la representación de vector del primer documento, si en lugar de utilizar un pesado tf, hubiera utilizado una codificación "SMART" de tipo ntc, es decir, el peso w_{jk} del término t_k en el documento d_j está dado por

$$w_{jk} = \frac{tfidf(t_k, d_j)}{\sqrt{\sum_{s=1}^{m} tfidf(t_s, d_j)^2}}$$

   donde

$$tfidf(t_k, d_j) = tf(t_k, d_j) \cdot \log(\frac{n}{DF_{t_k}})$$

   y $tf(t_k, d_j)$ es el número de veces (frecuencia) que t_k ocurre en d_j, n es el número de documentos, m es el número de términos y DF_{t_k} es el número de documentos en que ocurre el término t_k.

2. En las 3 representaciones distribucionales vistas (DOR, TCOR y CSA), primero se obtenía una representación vectorial de los términos y luego se generaba una representación de los documentos haciendo una suma ponderada de los vectores de las palabras que aparecían en cada documento. Esto lleva a que la dimensionalidad de los vectores que representan a los documentos en cada caso, sea igual a la dimensionalidad de los vectores de los términos que le dieron origen. En base a las anteriores consideraciones, diga cual sería la dimensionalidad de un documento en cada uno de los enfoques distribucionales (DOR, TCOR y CSA) para los documentos del ejemplo. Para el caso de CSA, asuma que los documentos están etiquetados como pertenecientes a dos clases: "deporte" o "política".

3. Para cada una de las 3 representaciones distribucionales (DOR, TCOR y CSA), obtenga la representación de vector de los términos "pelota" y "partido". En el caso de CSA, asuma que los 3 primeros documentos están etiquetados como pertenecientes a la clase "deporte" y los últimos dos a la clase "política".

‘

Nota: En caso de requerirse una explicación más detallada de los métodos DOR y TCOR, en [2] se puede encontrar una buena descripción de los métodos. Más tarde, en [1] los autores las utilizaron en las representaciones de los documentos para la categorización de textos cortos. La descripción original del método CSA puede encontrarse en [3]. En la presentación disponible en `https://ccc.inaoep.mx/`

`~hugojair/Courses/RUSSIR/2_BagOfConcepts.pdf`, se da un buen pantallazo general de los métodos distribucionales y en particular de DOR, TCOR y CSA.

Parte C: Categorización y clustering de documentos

Ejercicio 6 (Evaluación de un clasificador de textos)

Suponga la siguiente tabla de contingencia (matriz de confusión), para un problema de "author profiling" donde un clasificador automático *clas*, categorizó los documentos del conjunto de prueba ("test set") en una de tres clases: adolescente (edad de 13-17), joven (23-27) y adulto (33-37):

clas (estimado)	*real* (experto/"ground-truth")			
		adolescente	joven	adulto
	adolescente	45	3	1
	joven	8	16	1
	adulto	4	2	20

Como se puede observar, por ejemplo, 45 documentos que fueron categorizados como adolescente por el clasificador, también lo fueron por el experto (hubo coincidencia/acierto). Sin embargo, 2 documentos categorizados como adulto por el clasificador, en realidad correspondían a la clase joven.

En base a esta matriz de confusión, determine:

1. El número total de documentos en el test set.
2. El total de documentos que el clasificador categorizó como joven.
3. El total de documentos rotulados como adolescente por el experto.
4. La exactitud (accuracy) del clasificador.

Ejercicio 7. Calcule ahora la precision (π_C) y recall (ρ_C) de cada clase C correspondiente a la matriz de confusión del ejercicio anterior. Observar que, por ejemplo, $\pi_{adolescente}$ y $\rho_{adolescente}$ pueden ser determinadas directamente de la matriz de confusión presentada en el ejercicio previo, o a partir de la matriz de confusión específica para la clase adolescente:

clas	*real*		
		adolescente	¬adolescente
	adolescente	45	4
	¬adolescente	12	39

En este caso, vemos que cuando consideramos específicamente a la clase adolescente tenemos, por ejemplo, 45 Positivos Verdaderos (PV), es decir, 45 documentos correctamente clasificados como adolescente. Por otra parte, se produjeron 4 Falsos Positivos (FP), es decir 4 casos incorrectamente clasificados como adolescente cuando en realidad no lo eran.

Notar que si quiciéramos obtener un valor de precision y recall de todo el clasificador, debemos realizar alguna forma de promedio de los valores de precision y recall obtenidos con cada una de las clases. En clasificación de textos, dos formas de promedio suelen ser utilizadas en estos casos: a) micropromediado (en inglés "microaveraging") denotados como $\hat{\pi}^{\mu}$ y $\hat{\rho}^{\mu}$ para precision y recall respectivamente y b) macropromediado (en inglés "macroaveraging") denotados como $\hat{\pi}^{M}$ y $\hat{\rho}^{M}$. El lector interesado puede obtener en el paper de Sebastiani [4] (página 38) una explicación detallada de estos conceptos.

Ejercicio 8. Rara vez precision y recall son consideradas en forma aislada. De ser así, siempre es posible encontrar clasificadores triviales que maximizan una a expensas de la otra. Una solución es encontrar medidas combinadas como la "F-measure" (medida F):

$$F = \frac{2\pi\rho}{\pi + \rho}$$

que es la media armónica de π y ρ. La medida previa, no es más que un caso particular (F_1) de la función F_β:

$$F_\beta = \frac{(\beta^2 + 1)\pi\rho}{\beta^2\pi + \rho}$$

para algún $0 \leq \beta \leq +\infty$. Usualmente $\beta = 1$, le cual da igual peso a π y ρ. Sin embargo, distintos β pueden ser utilizados dependiendo del problema. En los casos extremos, F_0 es π, mientras que $F_{+\infty}$ equivale a ρ.

Ejercicio: suponga una competencia de identificación de predadores sexuales, donde se tiene un conjunto de test que contiene 254 predadores.

En esta competencia, cada competidor (clasificador) recupera/clasifica una cantidad *Rec* de bloggers que él considera que son predadores. De esos bloggers recuperados, sólo una cantidad *Ac* son aciertos, es decir, eran efectivamente depredadores ($Ac \leq Rec$). Dada la siguiente tabla de resultados:

Participante	Official rank	*Rec*	*Ac*	Precision	Recall	F_1	$F_{0,5}$
Tito	?	186	183	?	?	?	?
Pepe	?	204	200	?	?	?	?
Tato	?	181	170	?	?	?	?
Poro	?	159	154	?	?	?	?

Obtenga la matriz de confusión de cada participante y complete las columnas faltantes, considerando que el ranking oficial se determina por el valor de $F_{0,5}$.

Ejercicio 9 (Clustering y medidas de similitud)

Introducción

Una de las componentes principales en el clustering (agrupamiento) de documentos es aquella encargada de determinar la *similitud* (parecido) entre dos documentos. Entre las distintas propuestas para realizar esta tarea se puede mencionar el uso del coeficiente de *Matching Simple* y de *Jaccard*, la *similitud coseno* y la *distancia de edición mínima* (distancia de Levenshtein). Las dos primeras se suelen utilizar en aquellos casos en que el documento se representa con pesado binario, la tercera (coseno) sirve para distintos pesados de los términos y la última (Levenshtein) es muy utilizada para medir la diferencia entre cadenas cortas de caracteres. Realizaremos por lo tanto una mínima práctica en este ejercicio con estas medidas, luego de repasar en cada caso algunos conceptos básicos sobre las mismas.

Ejercicio 9.a - coeficientes de *matching simple* y de *Jaccard*.

Cuando se busca medir la similitud entre dos objetos **p** y **q** representados como vectores binarios, se suele definir una matriz M del tipo:

		q 1	**q** 0
p 1		M_{11}	M_{10}
p 0		M_{01}	M_{00}

donde M_{xy} = número de atributos donde **p** es x and **q** es y.

A modo de ejemplo, con dos objetos (vectores binarios) **p** y **q** con los siguientes valores:

$$\mathbf{p} = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$
$$\mathbf{q} = (0, 0, 0, 0, 0, 0, 1, 0, 0, 1)$$

la matriz M correspondiente sería:

		q 1	**q** 0
p 1		0	1
p 0		2	7

1. En base a la información mostrada previamente, calcule la similitud entre los puntos **p** y **q** utilizando el coeficiente de *matching simple* y el coeficiente de *Jaccard*, considerándose que estos coeficientes se definen de la siguiente manera:

$$mat - simple = \frac{M_{11} + M_{00}}{M_{01} + M_{10} + M_{11} + M_{00}}$$

$$jacc = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

2. Suponga ahora que se tiene una representación matricial (binaria) de los documentos con 15 atributos (términos) $A_1 \ldots A_{15}$ y las 4 instancias (documentos) que se muestran a continuación:

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}
i_1	1	1	0	0	1	0	1	0	1	1	0	0	1	1	1
i_2	0	0	0	1	1	1	0	1	0	1	0	0	1	0	1
i_3	1	0	0	1	1	1	1	0	0	1	0	1	0	1	1
i_4	0	1	1	0	0	1	1	1	1	0	0	0	0	0	1

Diga cual es la instancia (documento) más cercana a la instancia i_2, de acuerdo al coeficiente de *Jaccard*.

Ejercicio 9.b - similitud coseno.

Una de las medidas de similitud más populares, especialmente cuando se comparan vectores que representan documentos, es la medida de similitud *coseno*, definida como:

$$cos(\mathbf{p}, \mathbf{q}) = \frac{\mathbf{p} \cdot \mathbf{q}}{\|\mathbf{p}\|\|\mathbf{q}\|}$$

donde $\mathbf{p}$ y $\mathbf{q}$ son vectores numéricos arbitrarios, $\cdot$ es el producto escalar (interior) entre vectores, $\mathbf{p} \cdot \mathbf{q} = \sum_{k=1}^{n} p_k q_k$, y $\|\mathbf{p}\|$ es la longitud (magnitud) del vector $\mathbf{p}$, $\|\mathbf{p}\| = \sqrt{\sum_{k=1}^{n} p_k^2} = \sqrt{\mathbf{p} \cdot \mathbf{p}}$.

1. Calcule la similitud coseno de los dos objetos de datos $\mathbf{p}$ y $\mathbf{q}$ que se muestran a continuación, que podrían representar vectores de documentos:

$$\mathbf{p} = (3, 2, 0, 5, 0, 0, 0, 2, 0, 0)$$
$$\mathbf{q} = (1, 0, 0, 0, 0, 0, 0, 1, 0, 2)$$

Ejercicio 9.c - distancia de edición mínima (Levenshtein)

La distancia de edición mínima entre dos cadenas de caracteres es el mínimo número de *operaciones de edición* (operaciones como *inserción*, *eliminación* y *substitución*) que se necesitan para transformar una de las cadenas en la otra.

A modo de ejemplo, perro está a una distancia de Levenshtein de 1 de pero ya que requiere una operación de eliminación (de una de las 'r') y una distancia de Levenshtein de 2 de perras ya que requiere de una substitución ('o'por 'a') y una inserción (la 's'del final).

Describa cual sería la distancia de Levenshtein entre las palabras INTENTION y EXECUTION justificando el resultado en base a las operaciones involucradas. Usted puede chequear si su resultado es correcto utilizando el calculador de distancias de Levenshtein disponible en

http://www.convertforfree.com/levenshtein-distance-calculator/. Puede leer más sobre este tipo de distancias en la página 21 del capítulo 2 del libro de Jurasky https://web.stanford.edu/~jurafsky/slp3/2.pdf.

Ejercicio 10 (Evaluación de agrupamientos: Coeficiente de Silueta)

Suponga que con los siguientes datos (uni-dimensionales) $\{1, 5, 9, 20, 25, 30, 35, 52, 54, 58, 60\}$ un algoritmo de clustering ha generado los siguientes grupos:

$$G_1 = \{1, 5, 9, 20\}$$
$$G_2 = \{25, 30, 35\}$$
$$G_3 = \{52, 54, 58, 60\}$$

1. Estime el coeficiente de silueta para el objeto 5 ($s(5)$) del grupo G_1 y para el objeto 20 del mismo grupo.
2. ¿Qué sucede si el objeto 20 es cambiado del grupo G_1 al G_2? Para determinar eso, recalcule nuevamente los valores de $s(5)$ y $s(20)$ con esta nueva configuración de los grupos.
3. En base a los valores obtenidos en los dos puntos anteriores, ¿qué puede decir sobre la pertenencia o no del objeto 20 al grupo G_1?

Nota: más información sobre el coeficiente de silueta puede ser accedida en la página 581 del capítulo de clustering de Tan, Steinbach y Kumar (https://www-users.cs.umn.edu/~kumar001/dmbook/ch7_clustering.pdf)

Ejercicio 11 Aplicaciones (opcional)

En el artículo "Vector-based word representations for sentiment analysis: a comparative study" (disponible en http://sedici.unlp.edu.ar/handle/10915/56763) realizamos un estudio preliminar de distintas representaciones de documentos en un problema particular de análisis de sentimiento. Describa cuales fueron las representaciones utilizadas, algoritmos utilizados, resultados obtenidos, etc y dé su opinión sobre este trabajo en base a los conceptos vistos en el curso. Si existe otro tipo de aplicación/trabajo de su interés, puede utilizar ese artículo si lo desea realizando un análisis similar.

Referencias

[1] J. M. Cabrera, H. J. Escalante, and M. Montes-y Gomez. Distributional term representations for short-text categorization. *LNCS. Computational Linguistics and Intelligent Text Processing. CICLing 2013.*, 7817, 2013.

[2] A. Lavelli, F. Sebastiani, and R. Zanoli. Distributional term representations: An experimental comparison. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pages 615–624, New York, NY, USA, 2004. ACM.

[3] Z. Li, Z. Xiong, Y. Zhang, C. Liu, and K. Li. Fast text categorization using concise semantic analysis. *Pattern Recognition Letters*, 32(3):441–448, Feb. 2011.

[4] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, Mar. 2002.