Text Mining

Diego Garrido

Departamento de Ingeniería Industrial
Universidad de Chile

18 de Octubre de 2018

Agenda

1. Representar un documento como un vector.
2. Procesamiento de texto.
3. Recuperación de información
4. Modelamiento de tópicos.

¿Por qué los datos no estructurados son tan importantes?

Las empresas están tomando decisiones solo con el 20% de la información a la que tienen acceso, ya que el 80% de su información es no estructurada y no se puede utilizar completamente. Las compañías han tratado de darle sentido a los datos no estructurados por años, pero solo un 78% afirma que tienen poca o nula información de sus datos no estructurados [1].

Fuentes de datos no estructurados en las empresas:

1. Medios de comunicación social. Ej: mención de una compañía o un producto en redes sociales.
2. Fuentes internas. Ej: Presentaciones, material de marketing y ventas, informes, correo electrónico, etc.
3. Contenido generado por el cliente. Ej: Comentarios en línea, historial de navegación, correos electrónicos a un equipo de soporte e incluso llamadas telefónicas al servicio al cliente.

[1] http://fredrikstenbeck.com/unstructured-data-important/

Embedding

¿Cómo trabajamos con datos no estructurados?

Embedding: Llevar datos a una representación númerica. Los modelos trabajan con representaciones númericas!. Además, con vectores tenemos nociones de distancia y podemos calcular similitud (o disimilitud) entre vectores.

En *text mining* existen dos tipos de *Embedding*, basados en frecuencia y en predicción. Algunas *keywords*:

1. **Documento (D)**: se refiere a una observación de tipo texto. Ej: comentario, relato, email, etc.
2. **Corpus (C)**: colección de documentos, corresponde al conjunto de datos.
3. **Vocabulario (V)**: corresponde a un conjunto de términos (tokens únicos normalizados) dentro del corpus que sobrevivieron a un procesamiento.
4. **Tokenization**: dividir el texto en entidades significativa llamadas tokens, si cada token es una palabra se habla de unigrams, si cada token son pares de palabras se habla de bigrams, tres palabras son trigrams, así hasta n-grams. En términos más computines es pasar de un string a una lista de strings. Ejemplo de unigrams: *'Buenos días estudiantes buenos' = ['Buenos', 'diás', 'estudiantes', 'buenos']*

Vector Space Model es un modelo que representa un documento como un vector de términos, donde cada término es una dimensión del vector.

- Documentos con diferentes palabras y largos viven en el mismo espacio.
- Estas representaciones son además llamadas **Bag of Words** (o bolsa de palabras), donde un documento es representado por un conjunto no ordenado de sus términos.
- El valor en cada dimensión is un peso que presenta la relevancia del término i del vocabulario V en el documento d.

$$d^{\rightarrow} = (w_1, \ldots, w_{|V|})$$

- Bag-of-words no representa la semántica de los documentos, palabras que son sinónimos por ejemplo son consideradas como términos diferentes, además el orden de las palabras dentro del documento se pierde.

- **Term Frequency (TF)**: También llamada Count Vector, el peso asociado a cada término del vocabulario dentro de un documento viene dado por la frecuencia del término dentro del documento (tf_i).
- **Term Frequency-Inverted Document Frequency (TF-IDF)**: Esta transformación consiste en normalizar la frecuencia de cada término tf_i de un documento por $idf_i = log(\frac{N}{n_i})$, donde n_i es las veces que el término ha aparecido al menos una vez en el total de documentos y N es el número de documentos, siendo el peso para el término i el siguiente $w_i = tf_i \times idf_i$.

Doc1: I like R.
Doc2: I like Python.

Term Frequency

	Doc1	Doc2
I	1	1
like	1	1
Python	0	1
R	1	0

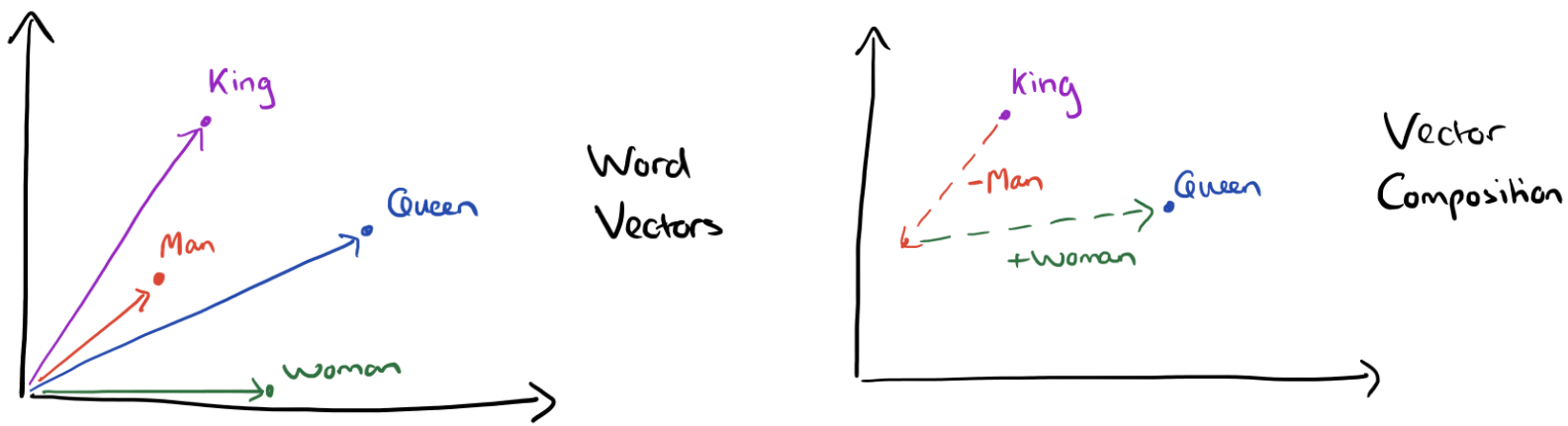
IDF

	IDF
I	0
like	0
Python	1
R	1

TF-IDF

	Doc1	Doc2
I	0	0
like	0	0
Python	0	1
R	1	0

- Los *Embedding* basados en predicción son principalmente arquitecturas de redes neuronales que "aprenden" la representación subyacente de las palabra, capturan la "semántica" de las palabras en una representación vectorial densa de baja dimensión (*word embedding*) y a partir de estos vectores se puede construir representaciones vectoriales de un texto.
- El típico ejemplo que se utiliza para mostrar su capacidad semántica es el de analogías, el cual consiste por ejemplo, en tomar la palabra rey, la palabra hombre y la palabra mujer, entonces, nos guastaría reponder la siguiente pregunta, hombre es a rey como mujer es a:



(a) **(b)**

Más ejemplos de analogías: `https://github.com/uchile-nlp/spanish-word-embeddings/blob/master/examples/Ejemplo_WordVectors.md`

Embedding basados en predicción [Mikolov et al., 2013].

1. **Skip – Gram model** : Se basa en la hipótesis que las palabras que aparecen en contexto similares tienen representaciones vectoriales similares, donde el contexto esta definido por las palabras vecinas, por ejemplo, por una ventana de 10 palabras, 5 palabras a la izquierda y 5 a la derecha. Skip-Gram entrena el contexto contra la palabra, preguntando, "dada esta única palabra, ¿cuáles son las otras palabras que probablemente aparecerán cerca de ella al mismo tiempo?"
2. **CBOW (Continuous Bag of words)**: El modelo CBOW entrena cada palabra en su contexto, preguntando, "dado este conjunto de palabras de contexto, ¿qué palabra faltante es probable que también aparezca al mismo tiempo?"

Algunos embeddings obtenidos en corpus en español:
`https://github.com/uchile-nlp/spanish-word-embeddings`.

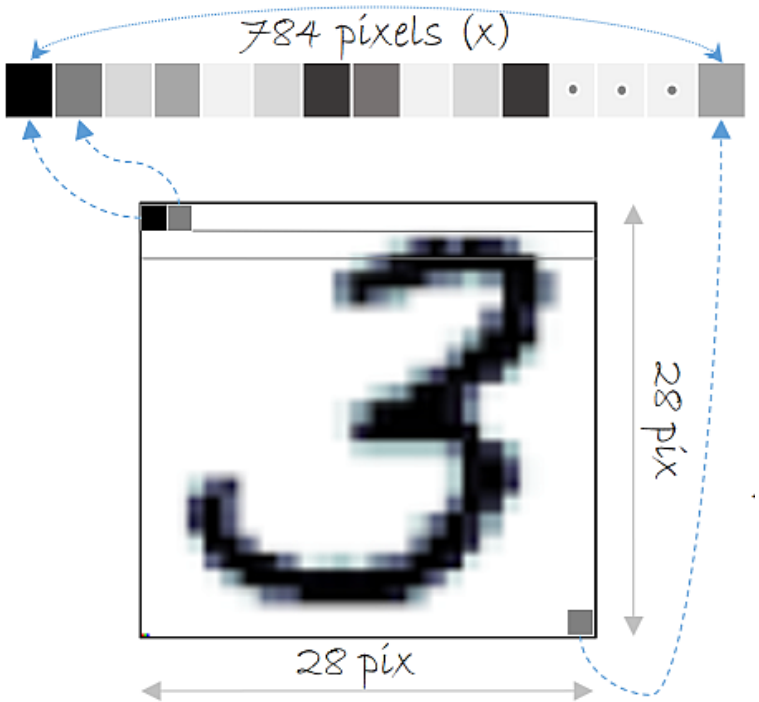
Comparación entre Embedding basado en frecuencia vs basado en predicción:

1. En textos no muy largo los *embedding* basados en predicción se alzan por sobre los basados en frecuencia, debida a su alta capacidad semántica.
2. Las principales desventajas de los *embedding* basados en frecuencia es que las representaciones obtenidas son *sparse*, es decir, se obtienen largos vectores con pocos valores no nulos, ya que la cantidad de palabras diferentes fácilmente puede sobrepasar los miles de palabras. Además, no permite relaciones semánticas complejas debido a que las palabras son intercambiables.
3. Las principales desventajas de los *embedding* basados en predicción es que las representaciones obtenidas para textos muy largos no son muy buenas, ya que para obtener una representación vectorial de un texto implica utilizar funciones de agregación sobre vectores de palabras y cuando más largo es el documento la influencia de cada palabra se pierde. Ej: $\vec{V_d} = \frac{\sum_{i=1}^{N} \vec{v_i}}{N}$, para el documento d que tiene N palabras, donde el vector de la palabra $i - ésima$ es $\vec{v_i}$.

Embedding

¿Y en imágenes?

- Una imágen de un solo canal (escala de grises) puede ser representada por un vector, donde cada columna es un píxel de la imagen y el valor en cada campo corresponde a un valor entre 0-255.
- Representación bastante pobre, ya que los píxeles en esta representación son independientes, lo cuál es antinatural, ya que los píxeles cercanos tienen alta dependencia, ¿Sucede lo mismo con la matriz término-documento?



Procesamiento

- El vocabulario español-latino tiene 88.000 palabras según la RAE de las cuáles 20.000 palabras activas y unas 40.000 pasivas se usan normalmente [2]. Nos enfrentamos a un problema de alta dimensionalidad!
- El objetivo del procesamiento en *textmining* es reducir el vocabulario, eliminando aquellas palabras irrelevantes y homologando palabras con un significado similar.
- **Tokenization**: dividir el texto en entidades significativa, por ejemplo en palabras unitarias (unigram), pares de palabras (bigram), tres palabras (trigram), etc. Cada entidad representa una columna en la matriz término documento. Pasar de un *string* a una lista de *strings*. Ej: *'Buenos días estudiantes buenos'* = *['Buenos', 'diás', 'estudiantes', 'buenos']*

Buenas prácticas de procesamiento:

1. Eliminar caracteres no alfa-numéricos: @_#%$/(?
2. Corrección de Ortografía. Las palabras con mala ortografía suelen tener baja frecuencia.
3. Eliminar números, urls, correos (depende del contexto).
4. Identificar palabras claves de ser necesario, por ejemplo, "Juegos Olímpicos" debiese ser considerado como un token "juegos_olimpicos".

[2] https://elpais.com/diario/2010/11/27/babelia/1290820336_850215.html

Procesamiento

Algunas técnicas que podemos aplicar para reducir el vocabulario:

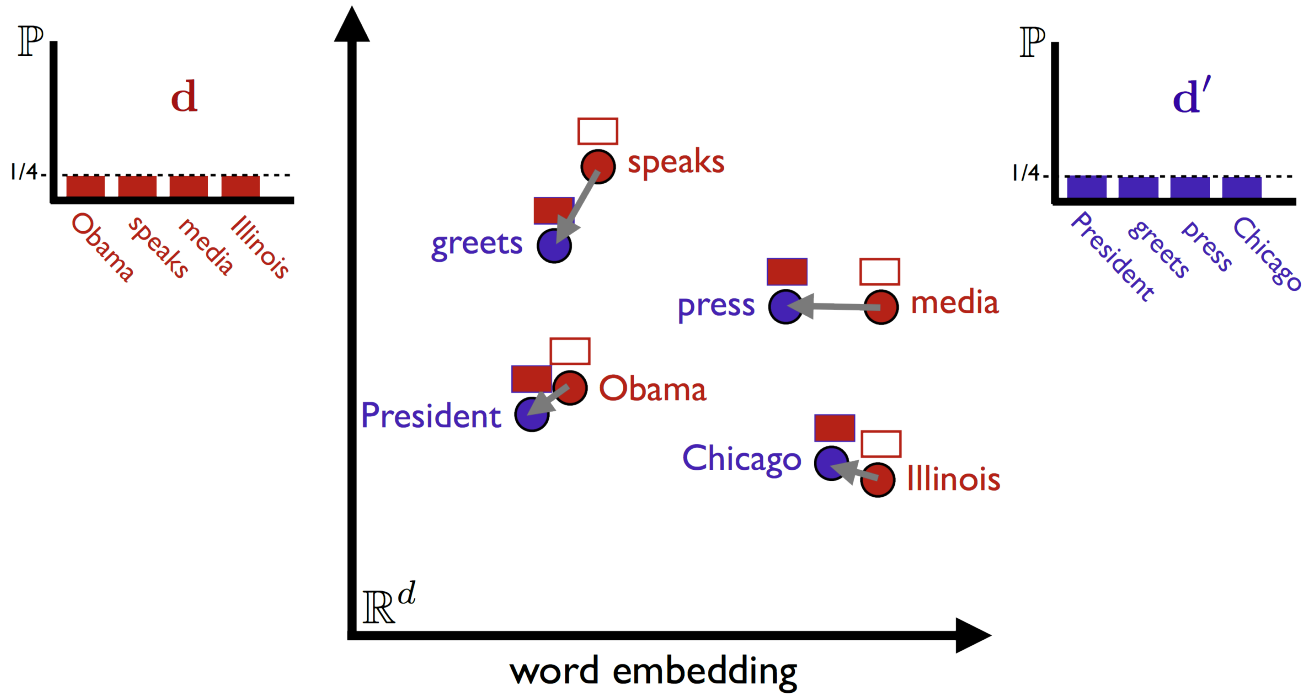
1. **Stop-words**: palabras que aportan poca información, por ejemplo: artículos, preposiciones, conectores.
2. **Stemming**: Llevar las palabra a su raíz gramatical.
3. **Lemmatization**: Llevar las palabras a su lema.
4. **Filtrar palabras de baja frecuencia**, por ejemplo menor a 10.
5. **Filtrar palabras con alta frecuencia** (stopwords contextuales), por ejemplo aquellas palabras que aparecen en todos los documento. En este caso una transformación TF-IDF genera una columna de ceros, esto implica nula o poca heterogeneidad por ende eliminar.
6. **Part of Speech Tagging (POST)**: etiquetado gramatical. Se suele utilizar en selección de atributos, por ejemplo dejando solo los sustantivos, pues estos son los que tienen la mayor parte de la información en una oración.
7. Usar **técnicas de selección de atributos**, por ejemplo, mutual information, chi-square, etc.

- **Information Retrieval**: ciencia de búscar información en una colección de fuentes de información. En este caso estamos interesados en búscar documentos similares dado un documento de búsqueda, para esto, primero necesitamos entender cómo podemos comparar de documentos. Algunas medidas de distancia y similitud:
  1. distancia Minkowski:
  $$L_p(x, y) = (\sum_{i=1}^{M} |x_i - y_i|^p)^{\frac{1}{p}}$$
  2. Similitud coseno:
  $$cos(x, y) = \frac{x \cdot y}{\|x\| \|y\|}$$
  3. Similitud euclideana:
  $$s_{L_2} = \frac{1}{1 + L_2(x, y)}$$
- En el caso de las medidas de distancia mientras mayor es la distancia entre dos vectores mayor es la disimilitud y en el caso de que la distancia es 0 se dice que los vectores son idénticos, por otro lado en las medidas de similitud como la coseno va entre [-1, 1], donde 1 significa que dos vectores son idénticos, -1 que son opuestos.

- Con el modelo *bag of words* obtenemos representaciones vectoriales de los documentos que podemos utilizar para calcular similitud (o disimilitud) entre documentos.
- Desventajas:
  1. Vectores sparse, dos vectores de documentos son casi ortogonales, por lo que la similitud puede ser baja.
  2. No captura distancias entre palabras, palabras que son sinónimos ocupan un token diferente. Un ejemplo del caso anterior es cuando tenemos dos documentos que no tienen palabras en común, pero dicen lo mismo, solo que uno ocupa sinónimos de todas las palabras del otro, los vectores serían ortogonales y la similitud coseno es 0.

- Otra solución es agregar los word embedding y utilizar este descriptor para el cálculo de similitudes, en general entrega mejores resultados que *bag of words*, pero no es robusto a documentos largos ya que al agregar demasiadas palabras se empieza a perder la semántica y los vectores comienzan a parecerse mucho.

Word Mover's Distance (WMD) permite medir similitud (o disimilitud) entre documentos, utiliza dos ideas, la representación bag-of-words de un documento y word embedding. [Kusner et al., 2015]



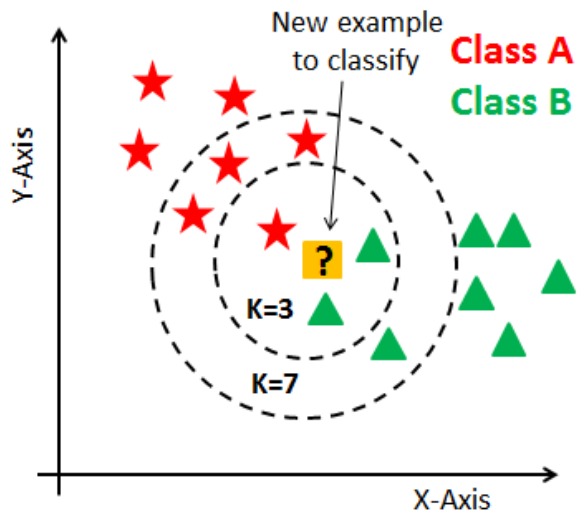
Ejemplo: `https://vene.ro/blog/word-movers-distance-in-python.html`

WMD busca minimizar el costo de transformar un documento a otro, para esto modelo el problema como un problema de flujo a costo mínimo (MCF).

$$\begin{aligned} \underset{T}{\text{minimize}} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\, T_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{m} T_{ij} = d_i \;\; i = 1, \ldots, m \\ & \sum_{i=1}^{n} T_{ij} = d_j^{'} \;\; j = 1, \ldots, n \\ & T_{ij} \geq 0, \;\; i = 1, \ldots, m, \;\; j = 1, \ldots n. \end{aligned}$$

- T_{ij} es el flujo que va de la palabra i del documento d a la palabra j del documento $d^{'}$, d_i es el peso que tiene la palabra i en la representación bag-of-words del documento d (frecuencia normalizada por el número de palabras), c_{ij} es el costo de mover una unidad de flujo por el arco (i, j), el costo entre palabras se mide como la distancia 'l1' o 'l2' entre los embedding de dichas palabras.
- La primera restricción indica que el flujo que se mueve de una palabra i del documento d a todas las palabras del documento $d^{'}$ debe sumar su peso.
- La segunda restricción significa que el flujo que se mueve de una palabra j del documento $d^{'}$ a todas las palabras del documento d debe sumar su peso.

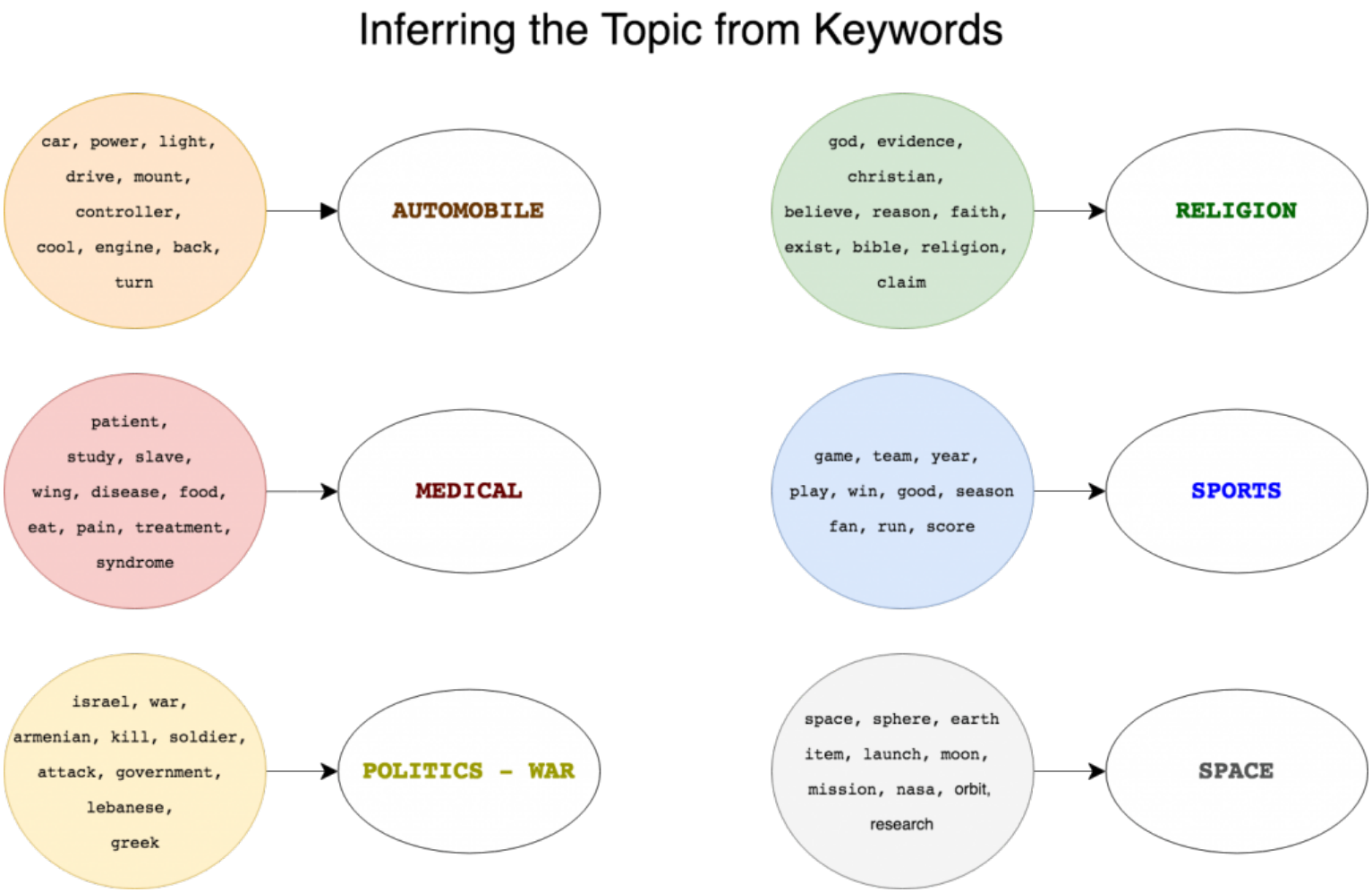
- Para evaluar la calidad de una técnica de recuperación de información se suele recurrir a aprendizaje supervisado y la tarea escogida es clasificación. Por ejemplo, si contamos con noticias etiquetadas, entonces, un buen método de recuperación de información dada una noticia como query debiese retornar dentro de las k noticias más similares noticias de la misma categoría.
- K-Nearest Neighbors es un modelo de clasificación el cual etiqueta una observación utilizando como input la etiqueta de las k observaciones más cercanas que pertenecen al conjunto de entrenamiento, por ejemplo, si tomamos una noticia del conjunto de test y el número de vecinos a buscar es $k = 10$ y el resultado de la búsqueda dió 8 noticias de la categoría Economía y 2 noticias de la categoría Nacional, entonces, la etiqueta para esta noticia es Economía.



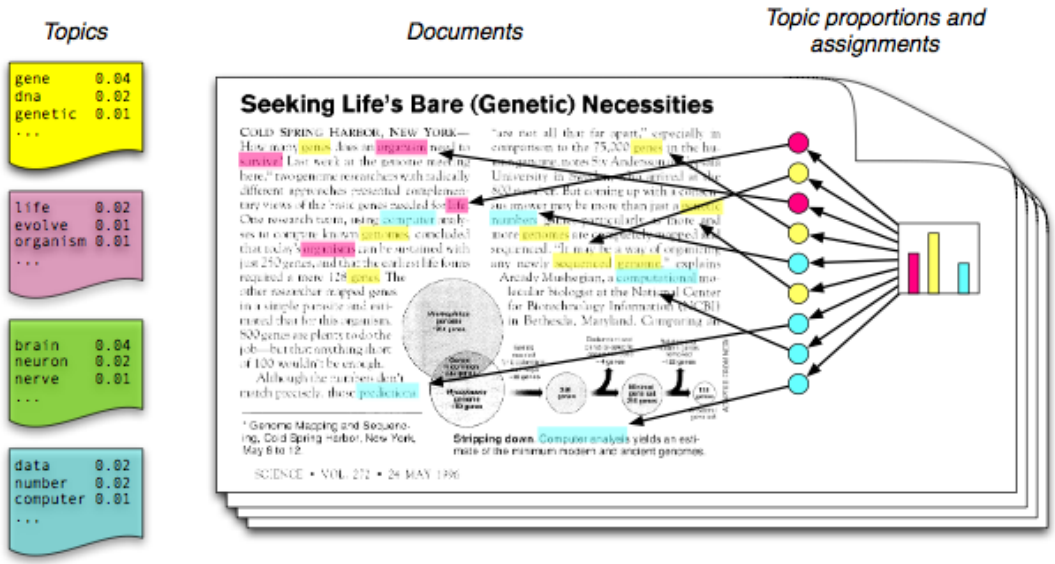
El modelamiento de tópicos contempla un conjunto de técnicas de aprendizaje no supervisado de clustering, busca **descubrir los tópicos (o temas) presentes en un corpus**.

- El modelamiento de tópicos sirve para organizar, buscar, indexar, explorar y comprender grandes colecciones de documentos.
- **Tópico**: Patrón repetitivo de términos co-currentes en un corpus, por ejemplo, se tiene el siguiente tópico, representado por sus **cuatro palabras más probables**, 'salud', 'médico', 'paciente', 'hospital', estas palabras sugieren el siguiente nombre para el tema: **'atención médica'**.
- Las palabras dentro de un documento y los documentos son tratados como intercambiables, por tanto, trabajan bajo un modelo **bag-of-words**.
- En general se debe especificar el número de tópicos a descubrir (K).

Los tópicos son interpretados por humanos a través de la lectura de las palabras más probables que lo componen. En la siguiente figura se tiene un ejemplo de etiquetado de tópicos a partir de sus 10 palabras más probables.



- Cada documento es generado por una mezcla de K tópicos, por ejemplo, para $K = 3$ la mezcla de tópicos puede ser $\theta_d = [0.1, 0.2, 0.7]$, en este caso el tópico 3 tiene más propabilidad de haber generado el documento o puede interpretarse que hay más presencia de ese tópico en el documento.
- Cada tópico es una distribución de probabilidad sobre un vocabulario V, por ejemplo, $\beta_k = [w_1^k, \ldots, w_{|V|}^k]$, donde $w_i^k \geq 0$ y $\sum_{i=1}^{|V|} w_i^k = 1$.



- Each **topic** is a distribution over words
- Each **document** is a mixture of corpus-wide topics
- Each **word** is drawn from one of those topics

- Una generalización multivariada de la distribución beta es la **distribución Dirichlet**, la cual tiene soporte sobre un símplice, definido por:

$$S_K = x : 0 \leq x_k \leq 1, \sum_{k=1}^{K} x_k = 1$$

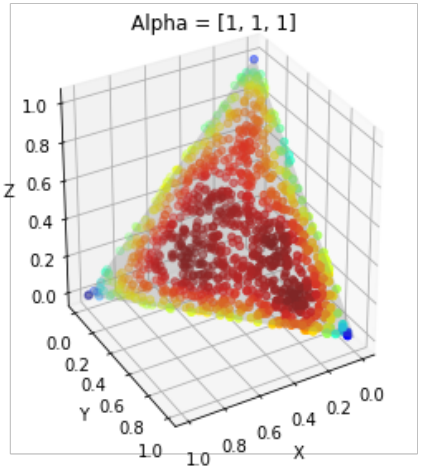
- Luego, su función de desnidad de probabilidad (pdf):

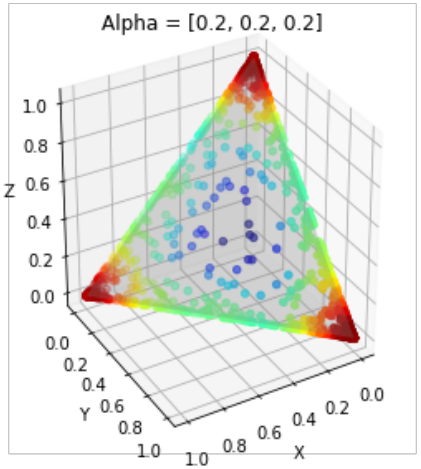
$$Dir(x|\alpha) = \frac{1}{B(\alpha)} \prod_{k=1}^{K} x_k^{\alpha_k - 1} \mathbb{I}(x \in S_K)$$

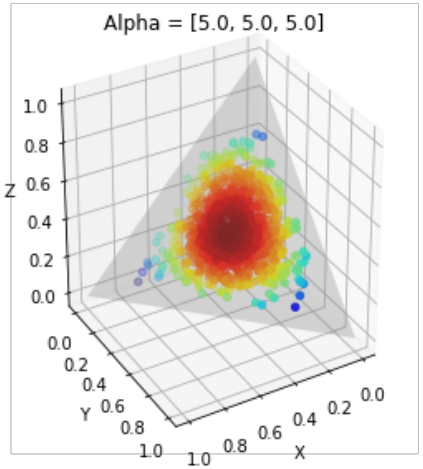
- La distribución Dirichlet es una distribución idónea para generar distribuciones de probabilidades categóricas, como la mezcla de tópicos que generó a un documento (θ_d) y un tópico (β_k). En general se asume simétria en los parámetros de la distribución, es decir, $\alpha_k = \frac{\alpha}{K}$.

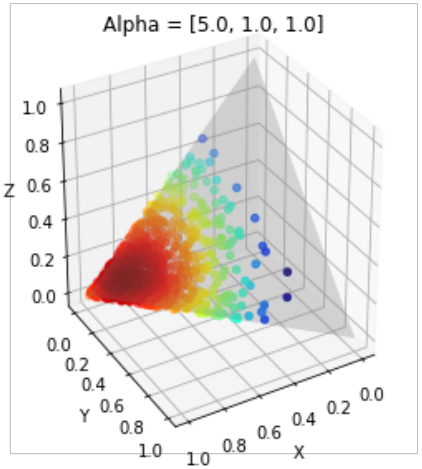
Topic Modelling

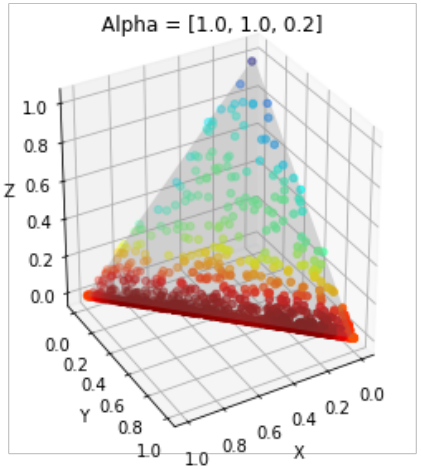
Dirichlet efecto de los parámetros

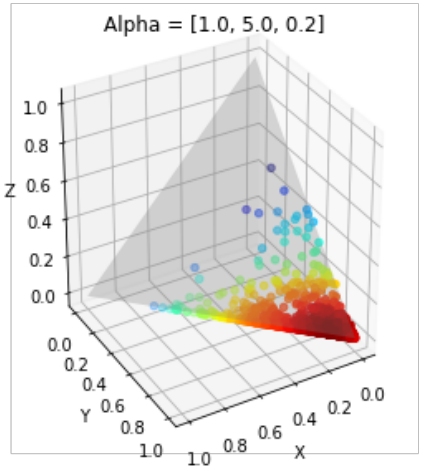




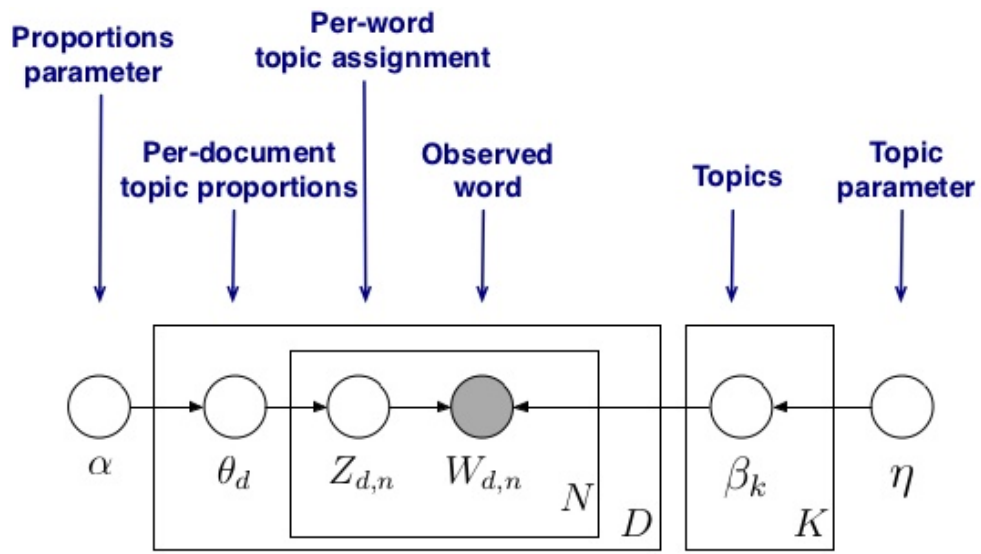








- Sean K tópicos, $\beta_{1:K}$ son distribuciones de probabilidad sobre un vocabulario fijo, dibujadas por una $Dirichlet(\frac{\eta}{|V|}1_{|V|})$.
- Para cada documento d del corpus D se asume que es dibujado por el siguiente proceso generativo:
  1. Dibujar una mezcla de tópicos $\theta_d \sim Dir(\frac{\alpha}{K}1_K)$.
  2. Para cada palabra:
     (a) Escoger un tópico $z_{d,n} \sim Mult(\theta_d)$.
     (b) Escoger una palabra $w_{d,n} \sim Mult(\beta_{z_{d,n}})$.



$$p(\beta, \theta, z, w|\alpha, \eta) = \prod_{k=1}^{K} p(\beta_k|\eta) \prod_{d=1}^{D} p(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} p(z_{d,n}|\theta_d) p(w_{d,n}|\beta_{1:K}, z_{d,n}) \right)$$

- La **perplexity** es una métrica de que tan bien un modelo probabilístico representa los datos, se cálcula a partir de la $\mathcal{L}(w)$ (log-likelihood) del modelo sobre un conjunto de datos.

- De un modelo generativo se espera que la probabilidad de generar un documento d observado sea alta (lo más cercano a 1), en el caso de LDA esta probabilidad viene dada por $p(w_d|\beta,\alpha)$, si esta probabilidad fuese 1 para todos los documentos la perplexity alcanzaría su valor mínimo que es 1, por tanto, un mejor modelo generativo tiene una perplexity más pequeña.

$$perplexity = 2^{-\mathcal{L}(w)},\ \mathcal{L}(w) = \sum_{d=1}^{D_{test}} log\left(p(w_d|\beta,\alpha)\right)$$

- El inconveniente de esta métrica es que suele tener un **comportamiento monótono** en el número de tópicos, es por esto que se suele utilizar el método del codo para decidir el número adecuado de tópicos.

Se ha demostrado que la perplexity y el juicio humano no están frecuentemente correlacionadas y que incluso algunas veces están sutilmente correlacionadas de forma negativa [Chang et al., 2009]. Esto suele pasar por que las palabras que tienen una alta frecuencia dentro de un tópico usualmente no son buenos describiendo una idea coherente.

- Realizaron un experimento a gran escala en la plataforma Amazon Turk. Para cada tópico, tomaron las cinco palabras más probables y agregaron una sexta palabra al azar. Luego, presentaron estas listas de seis palabras a las personas y les preguntaban cuál es la palabra intrusa.
- Si todas las personas preguntadas pudieran decir cuál es el intruso, entonces podemos concluir con seguridad que el tópico es bueno para describir una idea. Si, por otro lado, muchas personas identificaron otras palabras como intrusa, significa que no podían ver la lógica en la asociación de palabras, y podemos concluir que el tópico no era lo suficientemente bueno.

Como la perplexity no esta correlacionada con el juicio humano se han creado métricas de coherencia basadas en la idea de que palabras con significado similar tienden a ocurrir en contextos similares, entonces, los tópicos son considerados coherentes si la mayoría de sus palabras (o top N palabras) están relacionadas, es decir, que las palabras más probables de un tópico tiendan a co-ocurrir juntas.

Dentro de las médidas de coherencia exploradas la **coherencia CV** [Röder et al., 2015] logra la más alta correlación con el juicio humano, CV se cálcula a nivel tópico y la médida final es el promedio de los CV individuales, CV esta basada en cuatro partes:

1. Segmentación de las top N palabras del tópico en pares de palabras.
2. Calcula las probabiliades de ocurrencia entre pares de palabras.
3. Calcula una medida de confirmación que mide cuan fuertemente un conjunto de palabras soporta otro conjunto de palabras.
4. Calcula una medida de confirmación que mide cuan fuertemente un conjunto de palabras soporta otro conjunto de palabras.
5. Agregación de las medidas de confirmación individuales dentro de una medida de coherencia global.

Las visualizaciones en modelamiento de tópicos nos ayudan a responder tres preguntas:

1. **¿Cuál es el significado de cada tópico?**
2. **¿Cuán predominante es cada tópico?**
3. **¿Cómo se relacionan los tópicos entre sí?**

LDAvis [Sievert and Shirley, 2014] es una herramienta de visualización para responder estas preguntas. La herramienta a través de una visualización espacial responde la pregunta 2 y 3. Además para responder la pregunta 1 incorporan un gráfico de barras a la derecha del gráfico espacial que muestra las palabras más relevantes del tópico seleccionado dado un parámetro $\lambda \in [0, 1]$, entonces, la relevancia de la palabra w en el tópico k dado λ esta dada a través de la siguiente formula:

$$r(w, k|\lambda) = \lambda P(w|k) + (1 - \lambda)\frac{P(w|k)}{P(w)}, \lambda \in [0, 1]$$

Donde $P(w|k) = \beta_{k,w}$ es la probabilidad de que el término w sea generado por el tópico k, $P(w)$ es la probabilidad del término w en el corpus.

`Click aquí:  LDAvis example`

- Los tópicos $\beta_{1:K}$ sirven para resumir una gran colección de documentos, una persona a través de la lectura de las palabras más probables de cada uno obtiene una idea de los temas presentes en el corpus.
- La mezcla de tópicos de un documento θ_d sirve como un embedding del documento su uso puede ser:
  1. Reducción de dimensionalidad.
  2. Segmentación.
  3. Como *features* para un problema de aprendizaje supervisado.
  4. Recuperación de información.
- Sistemas de recomendación de tipo filtro colaborativo, donde las películas representan el vocabulario, los documentos a los usuarios y la frecuencia es una calificación en escala likert entre 1 y 5.

Algunos de los modelos de tópicos más relevantes.

Modelo	**Año**	**Citas**	**Paper**
Probabilistic Latent Semantinc Indexing (pLSI)	1999	5752	[Hofmann, 2017]
Non-negative Matrix Factorization (NMF)	2001	7835	[Lee and Seung, 2001]
Latent Dirichlet Allocation	2003	28408	[Blei et al., 2003]
Hierarchical Dirichlet Process (HDP)	2005	3848	[Teh et al., 2005]
Topic over Time (TOT)	2006	1344	[Wang and McCallum, 2006]
Dynamic Topic Model (DTM)	2006	2232	[Blei and Lafferty, 2006]
Correlated Topic Model (CTM)	2007	1074	[Blei et al., 2007]
Infinite Dynamic Topic Model (iDTM)	2012	166	[Ahmed and Xing, 2012]
Biterm Topic Model (BTM)	2013	506	[Yan et al., 2013]
Bursty Biterm Topic Model (BBTM)	2015	51	[Yan et al., 2015]

Referencias I

Ahmed, A. and Xing, E. P. (2012).

Timeline: A dynamic hierarchical dirichlet process model for recovering birth/death and evolution of topics in text stream.
arXiv preprint arXiv:1203.3463.

Blei, D. M. and Lafferty, J. D. (2006).

Dynamic topic models.
In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM.

Blei, D. M., Lafferty, J. D., et al. (2007).

A correlated topic model of science.
The Annals of Applied Statistics, 1(1):17–35.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003).

Latent dirichlet allocation.
Journal of machine Learning research, 3(Jan):993–1022.

Chang, J., Gerrish, S., Wang, C., Boyd-Graber, J. L., and Blei, D. M. (2009).

Reading tea leaves: How humans interpret topic models.
In *Advances in neural information processing systems*, pages 288–296.

Hofmann, T. (2017).

Probabilistic latent semantic indexing.
In *ACM SIGIR Forum*, volume 51, pages 211–218. ACM.

Kusner, M., Sun, Y., Kolkin, N., and Weinberger, K. (2015).

From word embeddings to document distances.
In *International conference on machine learning*, pages 957–966.

Lee, D. D. and Seung, H. S. (2001).

Algorithms for non-negative matrix factorization.
In *Advances in neural information processing systems*, pages 556–562.

Referencias II

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013).

Efficient estimation of word representations in vector space.
arXiv preprint arXiv:1301.3781.

Röder, M., Both, A., and Hinneburg, A. (2015).

Exploring the space of topic coherence measures.
In *Proceedings of the eighth ACM international conference on Web search and data mining*, pages 399–408. ACM.

Sievert, C. and Shirley, K. (2014).

Ldavis: A method for visualizing and interpreting topics.
In *Proceedings of the workshop on interactive language learning, visualization, and interfaces*, pages 63–70.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2005).

Sharing clusters among related groups: Hierarchical dirichlet processes.
In *Advances in neural information processing systems*, pages 1385–1392.

Wang, X. and McCallum, A. (2006).

Topics over time: a non-markov continuous-time model of topical trends.
In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433. ACM.

Yan, X., Guo, J., Lan, Y., and Cheng, X. (2013).

A biterm topic model for short texts.
In *Proceedings of the 22nd international conference on World Wide Web*, pages 1445–1456. ACM.

Yan, X., Guo, J., Lan, Y., Xu, J., and Cheng, X. (2015).

A probabilistic model for bursty topic discovery in microblogs.
In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.