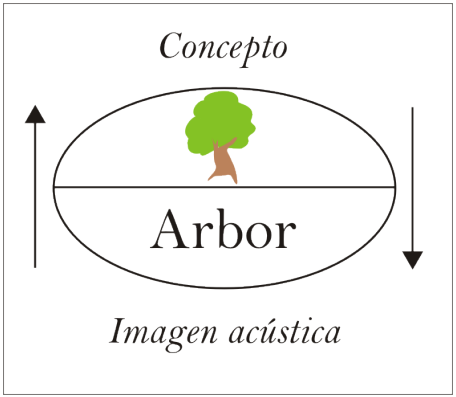
Búsqueda y Recuperación de Información en Textos

Víctor Mijangos de la Cruz

IV. Representación del significado

Representaciones del significado

¿Qué es el significado?



Una palabra se asocia a un **concepto**, que es la referencia de dicha palabra.

Existen diferentes **teorías del significado**. Una de ellas es la teoría **formalista**:

Teoría formal del significado

El significado de entidades lingüísticas emerge de la forma/estructura que tienen estas entidades.

Las tecnologías del lenguaje han adoptado esta perspectiva: extraer significado a partir de textos (forma).

Representaciones del significado

Existen diferentes formas de representar el significado computacionalmente. Algunas de estas son:

- **Lógicamente**: Utiliza fórmulas lógicas para traducir las oraciones del lenguaje natural en representaciones de su significado.
- **Relacionales**: Basados en las relaciones entre términos (como la relación *is_a*), caracterizan términos a partir de otros términos.
- **Modelo de espacio vectorial**: Representan la semántica de los términos en espacios vectoriales ($\mathbb{R}^d$).

Similitud semántica

Uno de los problemas esenciales de la recuperación de información es medir la **similitud semántica**:

Similitud semántica

La similitud semántica refiere a la "cercanía" o parecido que tienen dos documentos o términos en tanto a su significación (el tópico que tratan).

En las aplicaciones de RI es importante medir esta similitud:

Medida de similitud

Dado una colección $\mathcal{C} = \{d_1, ..., d_N\}$, una medida de similitud semántica es una función $sim : \mathcal{C} \times \mathcal{C} \to \mathbb{R}^+$ que determina la similitud semántica entre dos documentos.

Representaciones lógicas del significado

Representaciones de primer orden

La representación semántica puede realizarse con conceptos de la **lógica de primer orden**. Esta puede dividirse en dos **tipos de símbolos**:

Símbolos: Constantes (elementos sin argumentos) y relaciones (que toman uno o n argumentos, n es su aridad).

Operadores booleanos: AND $\wedge$, OR $\vee$, NOT $\neg$ y IF THEN $\rightarrow$.

Cuantificadores: Para todo $\forall$ y de existencia $\exists$.

Variables: Valores como x, y, z.

Términos: Los términos se componen combinando los elementos anteriores (p.ej. $x \wedge y$).

Sintaxis de porposiciones lógicas

Las fórmulas de las proposiciones lógicas se basan en los siguientes puntos:

- Si R es un símbolo de aridad n, y $t_1, ..., t_n$ términos, entonces $R(t_1, ..., t_n)$ es una fórmula.
- Si t_1 y t_2 son términos, entonces $t_1 = t_2$ es una fórmula.
- Si ϕ_1 y ϕ_2 son fórmulas, también lo son $\neg\phi_1$, $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$ y $\phi_1 \rightarrow \phi_2$.
- Si ϕ es una fórmula y x una variable, son fórmulas $\exists x\phi$ y $\forall x\phi$.

Semántica lógica

La aplicación de la semántica lógica busca asignar a las oraciones del lenguaje fórmulas lógicas que los describan. Por ejemplo a "Un gato entró a CU":

$$\exists x\Big(\textsc{Gato}(x) \wedge \exists y\big(y = \mathrm{CU} \wedge \textsc{entrar}(x, y)\big)\Big)$$

Podemos explicar esta fórmula como sigue:

- La relación $\textsc{Gato}$ que toma una variable (x es gato).
- Constante CU, nombre de entidad. Se asigna a la variable y.
- Una relación $\textsc{entrar}$ que relaciona x con y (x entró a y).

Descripción del vocabulario

Para transformar una oración en fórmula lógica se construye un **vocabulario**, este vocaubulario usa dos operadores:

- λ Abstrae la información (en variables que se utilizarán).
- @ expresa una aplicación funcional.

Forma	gato
Sintaxis	Sustantivo
Semántica	$\lambda x.\text{GATO}(x)$

Forma	entró
Sintaxis	Verbo
Semántica	$\lambda x.\text{ENTRAR}(x)$

Forma	CU
Sintaxis	Entidad
Semántica	$\lambda x.x = \text{CU}$

Forma	un
Sintaxis	Determinante
Semántica	$\lambda p.\lambda q.\exists x(p@x \wedge q@x)$

Mapeo de oración a fórmula lógica

Para pasar de la oración a la fórmula lógica se utiliza una β-**reducción**: reduce el lado derecho de @ insertándolo en el lado izquierdo, resolviendo por una variable λ.

Para la frase "un gato" tenemos:

$$\begin{aligned} un@gato &= \big(\lambda p.\lambda q.\exists x(p@x \wedge q@x)\big)@\big(\lambda x.\textsc{Gato}(x)\big) \\ &= \lambda q.\exists x(\lambda x.\textsc{Gato}(x)@x \wedge q@x) \\ &= \lambda q.\exists x(\textsc{Gato}(x) \wedge q@x) \end{aligned}$$

Esto puede conformar una nueva entrada determinada como:

Forma	un gato
Sintaxis	Frase nominal
Semántica	$\lambda q.\exists x(\textsc{Gato}(x) \wedge q@x)$

Mapeo de oración a fórmula lógica

Para conformar una oración como "un gato entró" se realiza la siguiente reducción:

$$\begin{aligned}\text{un gato@entró} &= \big(\lambda q.\exists x(\textsc{Gato}(x) \wedge q@x)\big)@\big(\lambda x.\textsc{Entrar}(x)\big)\\ &= \exists x(\textsc{Gato}(x) \wedge \lambda x.\textsc{Entrar}(x)@x)\\ &= \exists x(\textsc{Gato}(x) \wedge \textsc{Entrar}(x))\end{aligned}$$

De tal forma que podemos leerlo como "Existe x, tal que x es un gato y x entró".

Esto puede conformar una nueva entrada determinada como:

Forma	un gato entró
Sintaxis	Oración intransitiva
Semántica	$\exists x(\textsc{Gato}(x) \wedge \lambda x.\textsc{Entrar}(x)@x)$

Problemática de la perspectiva lógica

Las perspectiva lógica presenta algunas desventajas en su aplicación computacional:

- Describir un **vocabulario completo** es una tarea **exhaustiva** que requiere de un conocimiento profundo de la lengua.
- Están **orientados al lenguaje**; extenderlo a otras lenguas requiere modificaciones importantes.
- Son demasiado **rígidas** y no pueden lidiar fácilmente con ambigüedades.
- Su extensión para **capturar tiempo, modo** u otras características de la lengua complejiza el modelo.

Métodos relacionales de representación del significado

Relaciones entre términos

Se asume que existen **relaciones** entre instancias/términos y que existen **jerarquías** entre éstas.

Algunas de las **relaciones léxicas** que se asumen son:

- **Hiperónimo**: Es un término que describe a otro término (ej. animal es hiperónimo de mamífero).
- **Hipónimo**: Término conceptualmente más limitado (ej. mamífero es hipónimo de animal).
- **Sinónimo**: Dos términos que representan un significado idéntico (ej. blanco puede ser sinónimo de albo).
- **Antónimo**: Dos términos que presentan significados opuestos (ej. negro antónimo de blanco).
- **Meronimia**: Es una relación de contenido, implicada por "tener" (ej. llanta es merónimo de automóvil).

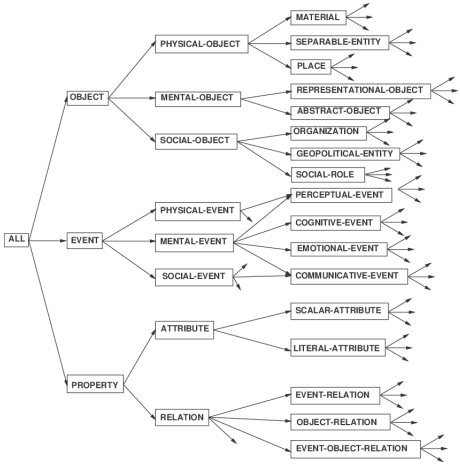
Representación en ontologías

Ontología

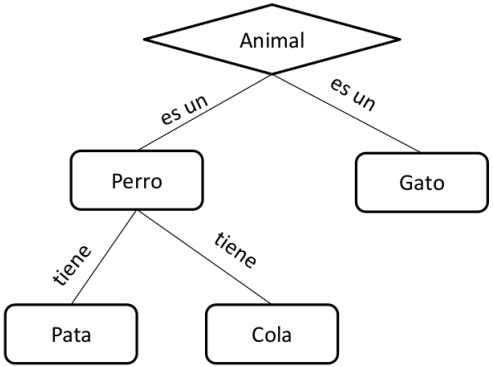
Una ontología es una gráfica que busca representar conocimiento a partir de relacionar dos términos w_1 y w_2 bajo una relación léxica.

Por ejemplo, se puede crear una ontología con **relaciones hiperónimo-hipónimo**:

$$animal \rightarrow mamfero \rightarrow felino \rightarrow gato$$



Componentes de una ontología



Generalmente, una ontología se compone de:

Conceptos: Son términos "abstractos" y generales que engloban una serie de instancias.

Instancias: Refieren a los objetos principales en los que se enfoca la ontología.

Relaciones: Son las relaciones que se establecen entre los conceptos y las instancias.

Tipos de ontologías

Según su **especificiadad** se puede hablar de:

- **Ontologías de alto nivel**: Son aquellas que buscan representar un conocimiento general del mundo.
- **Ontologías de tarea**: Describen conocimiento orientado a una tarea específica para resolverla.
- **Ontologías de dominio**: Describen conocimiento de un dominio específico (ej. de computación).
- **Ontologías de aplicación**: Conocimiento a la vez de dominio y de tarea particular, por medio de la especialización de los conceptos de las ontologías de dominio y de tareas.

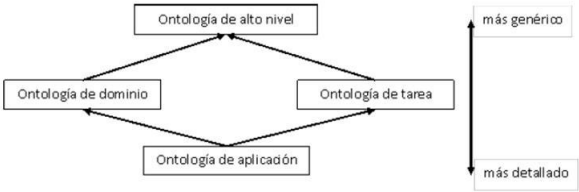


Fig. 1. Los cuatro tipos de ontologías según Guarino [9]

Algunas ontologías disponibles

Realizar una ontología es un trabajo muy costoso, pues requiere de mucho conocimiento especializado. Algunas ontologías que pueden consultarse son:

- **WordNet:** Se trata de una ontología que contiene términos en inglés, se puede consultar en https://wordnet.princeton.edu/. También es parte de la biblioteca NLTK.
- **DBpedia:** Es una ontología basada en la Wikipedia en inglés. Se puede consultar en https://dbpedia.org/ontology/.

Cálculo de similitud con ontologías

A partir de las ontologías podemos obtener medidas de similitud entre dos términos. La intuición es la siguiente:

Intuición de similitud

Dos términos serán más similares entre más hiperónimos compartan, así como entre menos hiperónimos difieran.

Para determinar el número de **hiperónimos que comparten** dos términos w_1, w_2, definimos:

$$common(w_1, w_2) = |\{w : w \rightarrow^+ w_1 \wedge w \rightarrow^+ w_2\}|$$

El símbolo $w \rightarrow^+ w_i$ indica que existe un camino iniciado por w que termina en w_i de longitud ≥ 1.

Cálculo de similitud con ontologías

La **profundidad** de un término w es el número de nodos que van desde la raíz de la ontología (el concepto más general) hasta w:

$$depth(w) = |path(w)|$$

donde $path(w)$ es el camino $w_0, ..., w$, w_0 raíz.

Si $w_1 = w_2$, entonces $common(w_1, w_2) = depth(w_1) = depth(w_2)$. De aquí:

$$2common(w_1, w_2) \leq depth(w_1) + depth(w_2)$$

Y ya que $common(w_1, w_2) \geq 1$ (al menos comparten el nodo raíz), podemos definir una similitud.

Similitud en ontologías

Similitud en ontologías

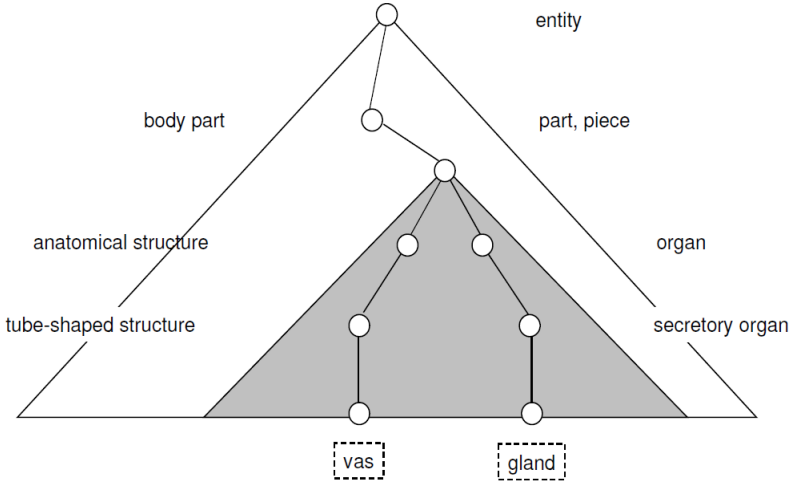
Dados dos términos w_1 y w_2 de una ontología, calculamos su similitud como:

$$sim(w_1, w_2) = 2\frac{common(w_1, w_2)}{depth(w_1) + depth(w_2)}$$

Esta similitud tiene las siguientes propiedades:

- $0 < sim(w_1, w_2) \leq 1$
- $sim(w_1, w_2) = 1$ si y sólo si $w_1 = w_2$

Similitud en ontologías: ejemplo



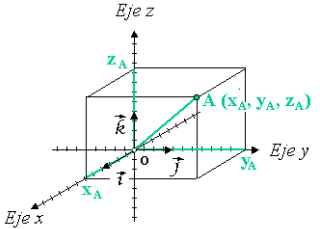
Modelos de espacio vectorial

Representación vectorial

Para la RI, un espacio vectorial refiere a un subconjunto (o todo) $\mathbb{R}^d$. Es decir, los elementos o vectores del espacio vectorial son d-tuplas:

$$x^T = \begin{pmatrix} x_1 & x_2 & \cdots & x_d \end{pmatrix}$$

Cada entrada del vector representa una coordenada, y por tanto, se pueden representar geométricamente a partir de un sistema coordenado:

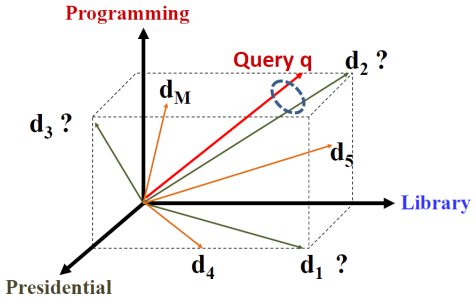


Sistema de Coordenadas Cartesianas Espaciales

Modelo de espacio vectorial

Modelo de espacio vectorial

El modelo de espacio vecotiral (en RI) refiere a la representación de términos y/o documentos en un espacio vectorial común.



Modelo del espacio vectorial

La idea de crear vectores es representar los términos y los documentos a partir de **información relevante** que nos indique cuándo dos elementos son similares.

La **similitud** corresponde a **cercanía** en el espacio vectorial.

El modelo de espacio vectorial tiene las siguientes ventajas:

- Se pueden realizar cálculos de similitud por medio de medidas del espacio vectorial.
- Se pueden hacer consultas determinando la similitud de un término con los documentos.
- Permite entregar algoritmos de aprendizaje de manera sencilla.

Algunos modelos de espacio vectorial

Dependiendo de la forma en que se construyan los vectores, podemos hablar de diferentes tipos de modelos de espacio vectorial.

- **Modelos de semántica distribucional**: Representan los términos a partir de sus relaciones con otros términos.
  - Matrices de co-ocurrencias: Frecuencias, PMI
  - Hyperspace Analogous to Language (HAL)
- **Modelos de bolsa de palabras (BOW)**: Representan a los documentos a partir de los términos, ignorando la estructura del lenguaje.
  - Modelos basados en TF-IDF
  - Modelo de semántica latente (LSA)
- **Modelos neuronales**: Aprenden las representaciones vectoriales a partir de redes neuronales.

Hipótesis distribucional

La construcción de modelos distribucionales se basa en la llamada hipótesis distribucional, atribuida al lingüísta Zellig Harris.

Hipótesis distribucional

Las palabras con carectirísticas semánticas similares se distribuyen en contextos similares.

De esta forma, podemos pensar en contextos como:

- Mi () es una mascota.
- El () me revisó los ojos.

Los espacios vacíos pueden ser llenados con palabras que se usan de manera similar, y por tanto tendrían un significado similar.

Coocurrencias en contextos

Un **contexto** puede ser:

- Una unidad lingüística, como la oración, el párrafo, el documento, etc.
- Una ventana de n términos a la izquierda y n términos a la derecha.

De esta forma, cada término puede estar representado por un vector de las palabras que comparten contexto con éste. Definamos la **coocurrencia** de los términos w_i, w_j en un contexto como:

$$c_{i,j} = |\{\mathcal{N}(w_i) : w_j \in \mathcal{N}(w_i)\}|$$

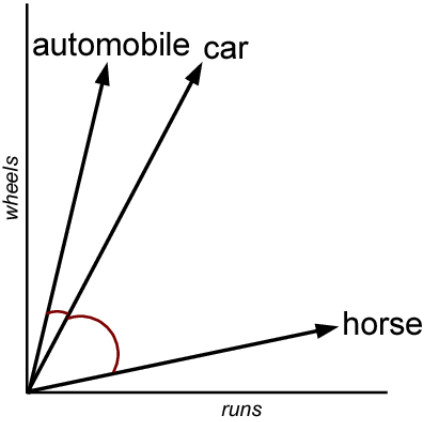
$\mathcal{N}(w_i)$ son los contextos de w_i (oraciones, documentos, ventanas). Definiremos una matriz de coocurrencias A como:

$$A = (a_{i,j}) = c_{i,j}$$

Matriz de coocurrencias

La idea es observar los contextos y ver cuando dos términos aparecen en un contexto similar, el número de contextos que compartan será su representación:

	runs	*wheels*
automobile	1	4
car	2	4
horse	4	1



Relevancia por cercanía

Otra aproximación a los DSMs consiste en tomar en cuenta la **relevancia** de una palabra en otra.

Esta relevancia se mide por la cercanía en la palabra. Si la palabra objetivo es w_0 y tenemos una **ventana** de $m \times m$ de contexto:

$$w_1 \ldots w_{m-1} w_m w_0 w_m w_{m-1} \ldots w_1$$

La relevancia se mide por la cercanía de la palabra w a w_0. Defínase esta cercanía como:

$$cl(w|w_0, c) = \text{cercanía de w a } w_0 \text{ en el contexto c}$$

La relevancia total de w dado w_0 se medirá como:

$$\phi(w, w_0) = \sum_c cl(w|w_0, c)$$

HAL

En base a la relevancia por cercanía, podemos definir un modelo de espacio distribucional conocido como **Hyperspace Analogous to Language** o HAL:

Modelo de HAL

El modelo de HAL (Hyperspace Analogous to Language) es un modelo de espacio disstribucional donde cada término w_0 se representa por sus relaciones de relevancia por cercanía con otros términos. En este caso, se tiene una matriz A, dada como:

$$A = (a_{i,j}) = \sum_{c} cl(w_i | w_j, c)$$

Cada renglón es un vector que representa al término w_j

Ejemplo de HAL

Supóngase que se tiene el siguiente texto:

El perro jugó con el gato

Si tomamos una ventana de 3×3 obtenemos las siguientes representaciones:

	el	perro	gato	jugó	con
el	0	4	3	4	4
perro	4	0	0	3	2
gato	3	0	0	1	2
jugó	4	3	1	0	3
con	4	2	2	3	0

Modelos basados en PMI

La medida de PMI (Pointwise Mutual Information) es un estándar para la representación en los modelos distribucionales.

Un modelo de DSM con PMI utiliza como entradas de los vectores los PMI entre el término objetivo y los otros términos:

$$A = (a_{i,j}) = PMI(w_i, w_j) = \log_2 \frac{p(w_i, w_j)}{p(w_i)p(w_j)}$$

Una versión general del PMI es el SPPMI$_k$, que define un rango para los valores positivos, y sólo toma estos valores:

$$PPMI_k(w_i, w_j) = \max\{0, \log_2 \frac{p(w_i, w_j)}{p(w_i)p(w_j)} - \log_2 k\}$$

Modelos basados en PMI

A partir del PPMI definimos el siguiente modelo:

Modelo PMI

El modelo de $PPMI_k$ es un modelo de espacio disstribucional donde cada término w_0 se representa por sus relaciones de relevancia por PMI con otros términos. En este caso, se tiene una matriz A, dada como:

$$A = (a_{i,j}) = \max\{0, \log_2 \frac{p(w_i, w_j)}{p(w_i)p(w_j)} - \log_2 k\}$$

Cada renglón es un vector que representa al término w_j

Cuando $k = 1$ tenemos que sólo se anula el rango, pues $\log 1 = 0$.

Maldición de la dimensionalidad y reducción

La **maldición de la dimensionalidad** señala que cuando los datos tienen dimensiones muy altas, éstos se vuelven difíciles de manejar, y los métodos que se aplican sobre ellos suelen ser inexactos.

Para lidiar con esta maldición, se han propuesto diferentes métodos de **reducción de la dimensionalidad**.

Varios de los métodos de reducción de la dimensionalidad se basan en la descomposición de matrices.

Matrices simétricas

Operador simétrico

Un operador A se dice que es simétrico si se cumple que:

$$\langle Ax, y \rangle = \langle x, Ay \rangle$$

En el caso de las matrices, se cumple que éstas son simétricas sii:

$$A = A^T$$

Si un operador (matriz) es simétrica, entonces admite una descomposición:

$$A = USV^T$$

Donde S es una matriz diagonal (de eigenvalores) y U, V son matrices (de eigenvectores).

Análisis de componentes principales

La descomposición en valores singulares busca una transformación $R : \mathbb{R}^d \rightarrow \mathbb{R}^m$, $m < d$, tal que $R^T R = Id$, de tal forma que tengamos:

$$\hat{R} = \arg\min_R \frac{1}{2} ||A - R^T RA||^2$$

Se puede comprobar que R corresponde a la matriz con columnas de m eigenvectores correspondientes a los eigenvalores más altos de la matrix $A^T A$ (que es simétrica):

Si $A^T A = USU^T$, entonces $R = U[: m]$. La reducción de dimensionalidad se hará como:

$$\hat{A} = RA$$

Composicionalidad

Un concepto importante en el manejo de unidades terminológicas es el de composicionalidad:

Composicionalidad

Dada una función de significado σ decimos que un elemento tiene significado composicional si el significado de este elemento es la suma de los significados de sus partes. Esto es:

$$\sigma(w_1 \cdot w_2) = \sigma(w_1) + \sigma(w_2)$$

De forma general, escogeremos una operación $\bigoplus$ (que no siempre tiene que ser la suma) de tal forma que:

$$\sigma(w_1 \cdot w_2) = \sigma(w_1) \bigoplus \sigma(w_2)$$

Tipos de composicionalidad

Algunas formas que se han propuesto para realizar la composición entre términos son:

Método	Formulación
Suma	$x + y$
Promedio	$\frac{x+y}{2}$
Producto	$x \odot$ y
AutoEncoder	AutoEncoder(x,y)

Un AutoEncoder es una red neuronal no supervisada; en este caso, puede tomar los dos vectores y obtener un vector nuevo.

Representación de documentos

Una forma de representar los documentos en vectores es a partir de capturar los términos que contienen.

Para esto, se puede usar la medida de TFIDF determinada como $tfidf(w_i, d_j) = tf_{i,j} \cdot idf_i$, donde:

$$idf_i = \log \frac{|D|}{|\{d : w_i \in d\}|}$$

Y la frecuencia de documento se puede definir como:

$$tf_{i,j} = \alpha + (1 - \alpha) \frac{f_{i,j}}{\max\{f_{k,j} : w_k \in d_j\}}$$

Vectores de TFIDF

El vector v_j que representa al documento d_j, se representará entonces como:

$$v_j = (tfidf_{i,j})$$

De tal forma que documentos que compartan mayor número de término tendrán mayor similitud entre sí. Esto permite **agrupar** los documentos.

A este tipo de modelos se les llama modelos de **Bolsa de Palabras**.

Relaciones términos-documentos

Una estrategia para representar tanto documentos como términos dentro de un espacio vectorial, es utilizar las relaciones que se dan entre vectores y documentos.

Se pueden utilizar las apariciones, pero es más común utilizar las **frecuencias** de un término dentro de un documento.

Si un término w_j a parece en un documento d_i con frecuencia $f_{i,j}$ podemos expresar una **matriz término documento** de la siguiente forma:

$$A = (a_{i,j}) = f_{i,j}$$

Correlaciones de la matriz

La matriz A es de tamaño $n \times m$, donde n es el número de documentos y m el número de términos.

Podemos ver que:

1. Si tomamos $A^T A$ tenemos una matriz de $n \times n$ que representa la correlación entre documentos.
2. Si tomamos AA^T tenemos una matriz de $m \times m$ que representa la correlación entre términos.

Descomposición de matrices

Ya que A^TA y AA^T son simétricas, podemos descomponerlas como:

$$A^TA = US^TSU^T$$

Y por otro lado:

$$AA^T = VSS^TV^T$$

De tal forma que podemos dar una aproximación a la matriz original de los datos como:

$$A \approx USV^T$$

Esta descomposición se conoce como **Descomposición en valores singulares** (SVD).

Reconstrucción de matrices

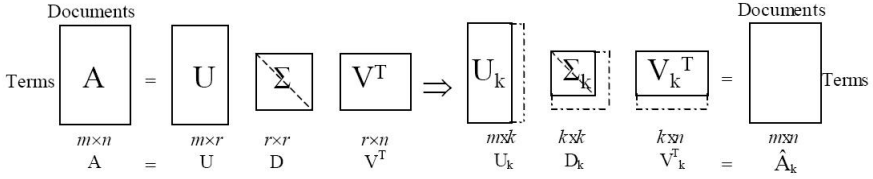
La matriz U representará a los **términos** y la matriz V a los **documentos**.

La matriz A puede reconstruirse a partir de los k **eigenvalues más relevantes** (de mayor valor):

$$\hat{A}_k = U_k S_k V_k^T$$

Esta reconstrucción tiene las ventajas de:

- Representa los elementos relevantes de los datos.
- Elimina ruido que no es relevante en los datos.



Consultas

Se pueden realizar consultas, considerando que:

- La matriz U_k representa los términos. Sea u_i el vector renglón que representa al término i.
- La matriz V_k representa los documentos. Sea v^j el vector que representa al documento d_j.
- La consulta mide la similitud entre términos y documentos.

Una consulta de los términos $t_1, ..., t_n$ se representa como:

$$q = \sum_{i=1}^{n} v_i$$

De tal forma que podemos definir el ranking:

$$rank(q, d_j) = \frac{|v^j \cdot q|}{||v^j|| ||q||}$$

Otros métodos de consulta

En el modelo de espacio vectorial, se pueden realizar consultas que buscan emular las **consultas booleanes**. Estas son:

- Consulta OR:

$$sim_{or}(q, d) = \left(\frac{\sum_i q_i^p d_i^p}{\sum_i q_i^p} \right)^{1/p}$$

- Consulta AND:

$$sim_{and}(q, d) = 1 - \left(\frac{\sum_i q_i^p (1 - d_i^p)}{\sum_i q_i^p} \right)^{1/p}$$

- Consulta NOT:

$$sim_{not}(q, d) = 1 - sim(q, d)$$

Modelo neuronal para RI

Frame Title

En RI, los sistemas de consultan incorporan los siguientes elements:

- Una **representación** de la **query**, que especifique la necesidad de información.
- Una **representación** de los **documentos**, que informe de la distribución de la información.
- Realizar **consultas** que relacionen los documentos con la query.

En el modelo de espacio vectorial las **representaciones** son **vectores**. En los **modelos neuronales** puede haber:

- Estimación de una **consulta** a partir de vectores de rasgos y una red neuronal profunda.
- Aprendizaje de las **representaciones** de términos y/o documentos a partir de una red neuronal.

Redes neuronales

Red neuronal FeedForward

Una red neuronal FeedForwad es un modelo gráfico que estima valores de salida dado entradas vectoriales. Esto es, es una función $f : \mathbb{R}^d \rightarrow Y$ tal que:

$$f(x) = \phi(W^{(K+1)} h^{(K)}(x) + b^{(K+1)})$$

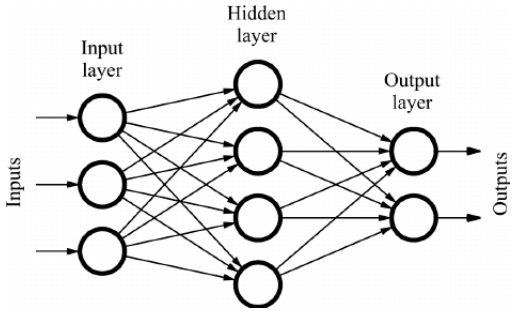
El modelo cuenta con **capas ocultas** $h^{(k)} : \mathbb{R}^{m_{k-1}} \rightarrow \mathbb{R}^{m_k}$, $k = 1, ..., K$ definidas como:

$$h(x) = g(W^{(k)} h^{(k-1)}(x) + b^{(k)})$$

Las funciones ϕ, g se conocen como funciones de activación y las matrices $W^{(k)}$ y los vectores $b^{(k)}$ son los pesos que se deben estimar.

Redes neuronales

Este tipo de redes se conceptualizan como una red, en donde las entradas de cada vector en cada capa representan las neuronas.



La entrada pasa a través de varias capas ocultas para obtener una salida adecuada.

Aprendizaje en redes neuronales

Para poder estimar la función dentro de una red neuronal deben aprenderse los pesos $W^{(k)}, b^{(k)}$, para toda capa oculta.

El proceso de estimar estos pesos se conoce como etapa de **entrenamiento**.

El entrenamiento depende de un conjunto de datos de entrada; por ejemplo, **vectores** que representan y una salida que es el valor a predecir (la similitud entre documento y término):

$$\mathcal{S} = \{(x, y) : x \in \mathbb{R}^d, y \in Y\}$$

Por ejemplo, x puede ser una representación vectorial de los documentos y los términos, mientras que y puede ser el valor de similitud a predecir.

Entrenamiento de las redes neuronales

Para entrenar las redes se define una **función de riesgo** que nos dice qué tanto se está equivocando nuestra red. Por ejemplo, podemos usar la función de riesgo:

$$R(\theta) = \frac{1}{2} \sum_x \sum_y (y - f(x))^2$$

Conocida como el **Error Cuadrático Medio** (o Mean Square Error, MSE).

Para minimizar esta función se hace uso del **gradiente descendiente**:

$$W_{i,j}^{(k)} \leftarrow W_{i,j}^{(k)} - \eta \nabla_{W_{i,j}^{(k)}} R(\theta)$$

Finalmente, el gradiente en cada capa de la red se obtiene por el método de **retropropagación**.

Algoritmo de backpropagation

Algorithm Backpropagation

procedure BACKPROPAGATION(R)
  Inputs: $R(\theta) \equiv R \circ \phi \circ h^{(K)} \circ h^{(K-1)} \circ \cdots \circ h^{(1)}$, ϕ es una red neuronal con pesos $\theta = \{W^{(K+1)}, W^{(K)}, ..., W^{(1)}\}$.
  $d_{out}(j) = \frac{\partial R}{\partial \phi_j(a)} \frac{\partial \phi_j(a)}{\partial a}$, $j = 1, ..., o$ — Inicialización
  $\frac{\partial R}{\partial W_{ij}^{(K+1)}} = d_{out}(j) h_i^{(K)}$, $j = 1, ..., o$
  for k **from** K **to** 1 **do**
    $d_k(j) = \frac{\partial h_j^{(k)}}{\partial a^{(k)}} \sum_q W_{j,q}^{(k+1)} d_{k+1}(q)$, $j = 1, ..., m_k$ — Inducción
    $\frac{\partial R}{\partial W_{i,j}^{(k)}} = d_k(j) h_i^{(k-1)}$, $j = 1, ..., m_k$ y $h^{(0)} = x$
  end for
  return $\frac{\partial R}{\partial W_{i,j}^{(k)}}$, para todo k, i, j
end procedure

Algoritmo de aprendizaje en redes FeedForward

procedure FitFeedForward(Red, R, T)
  Inicializa: $\theta = \{W^1, ..., W^{(K+1)}\}$ aleatoriamente.
  for t **from** 1 **to** T **do**
    for x, y **in** Batch(X, Y) **do**
      Forward:
        $$f(x) = \phi\left(W^{(K+1)} h^{(K)}(x) + b^{(K+1)}\right)$$
      Backward:
        $$\text{Loss} \leftarrow R(\theta; x, y)$$
        $$\frac{\partial R}{\partial W_{i,j}^{(k)}} \leftarrow \text{Backpropagation}(R(\theta; x, y))$$
        $$W_{i,j}^{(k)} \leftarrow W_{i,j}^{(k)} - \eta \cdot \frac{\partial R}{\partial W_{i,j}^{(k)}}$$
    end for
  end for
  return Red con pesos actualizados θ
end procedure

RI con redes neuronales

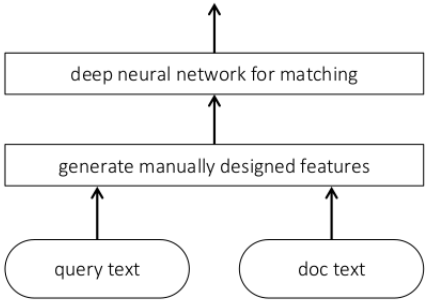
A partir de las redes neuornales se pueden realizar diferentes tipos de búsqueda. Aquí revisaremos 3 arquitecturas:

- Una red **supervisada** para estimar la **similitud** de un documento y un término a partir de vectores dados.
- Una red **supervisada** para estimar la **similitud** de un término y un documento, así como aprender sus **embeddings**.
- Una red **no-supervisada** para aprender los **embeddings** de los términos y documentos, que pueden usarse para cálculos de similitud.

Redes para aprender similitud

Supongamos que tenemos representaciones vectoriales de los documentos d_j, y de los términos t_j.

Podemos utilizar entonces una red neuronal para estimar la similitud entre ambos elementos:



Conjunto de entrenamiento

La entrada de la red serán los vectores del documento d_j y del término t_i, podemos **concatenar estos dos vectores**, de tal forma que:

$$x_{i,j} = [d_j : t_i]$$

Nuestra salida o clases serán los valores de relevancia para una consulta; por ejemplo, si el documento j satisface o no la necesidad de información por la consulta con el término t_i, denotamos esto $y_{i,j}$.

Por tanto, nuestro conjunto de entrenamiento será:

$$\mathcal{S} = \{(x_{i,j}, y_{i,j})\}$$

Estructura de la red

La red tomará como entrada el vector $x_{i,j}$ por lo que su primera capa será de la forma:

$$h^{(1)}(x) = g(W^{(1)}x + b^{(1)})$$

donde g es una función de activación ($\tanh$, ReLU, etc.)

Posteriormente se dfiniran K capas ocultas de la forma:

$$h^{(k)}(x) = g(W^{(k)}h^{(k-1)}(x)) + b^{(k)})k = 2, ..., K$$

Finalmente, la capa de salida tratará de predecir los valores $y_{i,j}$ (una sola neurona de salida).

$$y_{i,j} = \phi(W^{(K+1)}h^{(K)}(x) + b^{(K+1)})$$

La función ϕ puede ser lineal, o una función Sigmoide, que determina una probabilidad.

Predicción de relevancia

El modelo final es una red neuronal que dada las representaciones predice un valor de relevancia del término dentro de los documentos.

Sin embargo, vemos algunas desventajas en este modelo:

- Si las representaciones vectoriales no son adecuadas, las predicciones tampoco lo serán.
- Se requiere de una supervisión, que nos diga un valor de relevancia de un documento dado un término.

Aprendizaje de representaciones

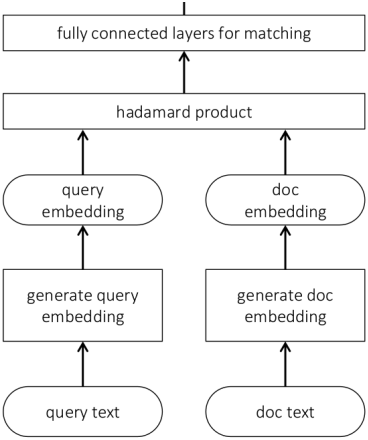
Una de las ventajas de las redes neuronales es que nos permiten aprender representaciones vectoriales sin tener que especificar rasgos específicos de los términos-documentos.

Una primera aproximación a la generación de embeddings es a partir de un **modelo supervisado**.

Este modelo aprende los **embeddings** a partir de **estimar la relevancia** entre la consulta y los documentos.

En este caso, los vectores d_j y t_i que representan al documento j y el términos i se aprenderán con la red neuronal.

Red supervisada



La red contará con los siguientes elementos:

- **Entrada:** Vectores de índices del documento con índice *j* y del término con índice *i*.
- **Capa de Embedding:** Capa lineal que tome los índices y los convierta en vectores de dimensión *d*.
- **Generación de vector término-documento:** Con los vectores de términos y de documentos y genere un sólo vector (por producto, suma o concatenación)
- **FeedForward:** La parte final de la red que tome el vector de entrada y obtenga una salida que prediga la relevancia del documento con la query.

Entrenamiento de la red

Esta red se entrenará, entonces con un conjunto de datos de entrenamiento de la siguiente forma:

$$\mathcal{S} = \{(i, j, s_{i,j}) : \text{término } i, \text{documento } j, \text{relevancia } s_{i,j}\}$$

El objetivo de la red es predecir la relevancia $s_{i,j}$ con base en d_j y t_i. Tenemos dos casos:

- Si la relevancia es un valor real, el riesgo es MSE: $R(\theta) = \frac{1}{2} \sum_{i,j} ||s_{i,j} - f(t_i, d_j)||$, $f(\cdot)$ es la red con capa lineal de salida.
- Si $s_{i,j}$ es una probabilidad, el riesgo es: $R(\theta) = -\sum_{i,j} p(s_{i,j}|t_i, d_j) \ln p(s_{i,j}|t_i, d_j)$, donde la probabilidad es predicha por la red (activación sigmoide).

Ventajas y desventajas del modelo

Las ventajas que este modelo presenta son que:

- No necesita representaciones vectoriales de los términos ni documentos.
- Se obtiene representaciones de de los términos y documentos de acuerdo a su relevancia.
- Se aprende una red capas de predecir la relevancia del término y el documento.

La principal desventaja del modelo es que:

- Se requiere conocer de antemano (al menos para un subconjunto) la relevancia de los documentos con los términos.

No supervisión

No siempre es fácil conocer la relevancia que esperamos dado un término y un documento. Por tanto, podemos hacer uso del **aprendizaje no supervisado**.

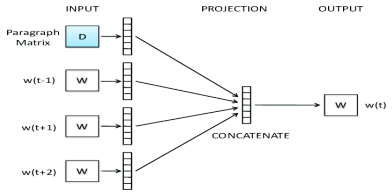
Un tercer modelo neuronal para RI consiste en **aprender los embeddings**, pero sin conocer previamente la relevancia; es decir **sin supervisión**.

Para hacer esto se hará uso de la estructura interna de los documentos, y las relaciones que guardan con los términos.

Aprendizaje de embeddings

Una opción es utilizar métodos de aprendizaje de embeddings tanto para términos como para documentos:

- **Word2Vec**: Es un modelo para aprendizaje de embeddings de términos basado en los contextos en que estos aparecen.
- **Doc2Vec**: Es un modelo neuronal para aprendizaje de embeddings de documentos basado en Word2Vec, este modelo aprende tanto embeddings para documentos, como embeddings para términos.



Aprendizaje de embeddings

En las representaciones vectoriales típicas, se crean matrices de términos-documentos que relacionan ambos elementos.

De forma similar podemos hacer que la red neuronal aprenda una matriz término documento, que factorice para obtener una salida:

$$p(t_i|d_j) = Softmax(U^i W_j)$$

Donde $U \in \mathbb{R}^{N \times d}$ es una matriz que representa los términos y $W \in \mathbb{R}^{d \times M}$ una matriz que representa los documentos.

Algoritmo para aprendizaje de embeddings

Algorithm Aprendizaje de embeddings

procedure FITNN($\mathcal{C} = \{d_1, ..., d_N\}$; η, T)
    $terms \leftarrow \bigcup_i d_i$
    $\mathcal{S} \leftarrow \{(d_j, t_i) : d_j \in \mathcal{C}, t_i \in terms\}$
    for t **from** 1 **to** T **do**
        for $d_j, t_i \in \mathcal{S}$ **do**
            $W_j \leftarrow$ columna j
            $\mathbf{p} \leftarrow \frac{\sum_k e^{U^k W_j}}{e^{UW_j}}$
            $U_{l,k} \leftarrow U_{l,k} - \eta(p_l - \delta_{l,i}) W_{j,k}$
            $W_{l,j} \leftarrow W_{l,j} - \eta \sum_q W_{l,q}(p_q - \delta_{q,i})$
        end for
    end for
    return W, embeddings de documentos, U, embeddings de términos
end procedure

Lecturas recomendadas

Blackburn, P., y Bos, J. (2003). "Computational semantics". *Theoria: An International Journal for Theory, History and Foundations of Science*, pp. 27-45.

Aranda G. y Ruiz, F. (2005). "Clasificación y ejemplos del uso de ontologáis en ingeniería de Software". *Workshop en Ingeniería del Software y Bases de Datos*.

Maynard, D. y Ananiadou, S. (2000). "Identifying Terms by their Family and Friends". *COLING*.

Baroni, M., Bernardi, R. y Zamparelli, R. (2014). "Frege in Space: A Program for Compositional Distributional Semantics". *Perspectives on Semantic Representations for Textual Inference.*, pp. 241-346.

Thomo, A. (2009). *Latent Semantic Analysis (Tutorial)*. http://www.ifp.illinois.edu/speech/speech_web_lg/slides/2017/thomo_svd_lsa_tutorial.pdf

Mitra, B. y Craswell, N. (2018). *An Introduction to Neural Information Retrieval*.