

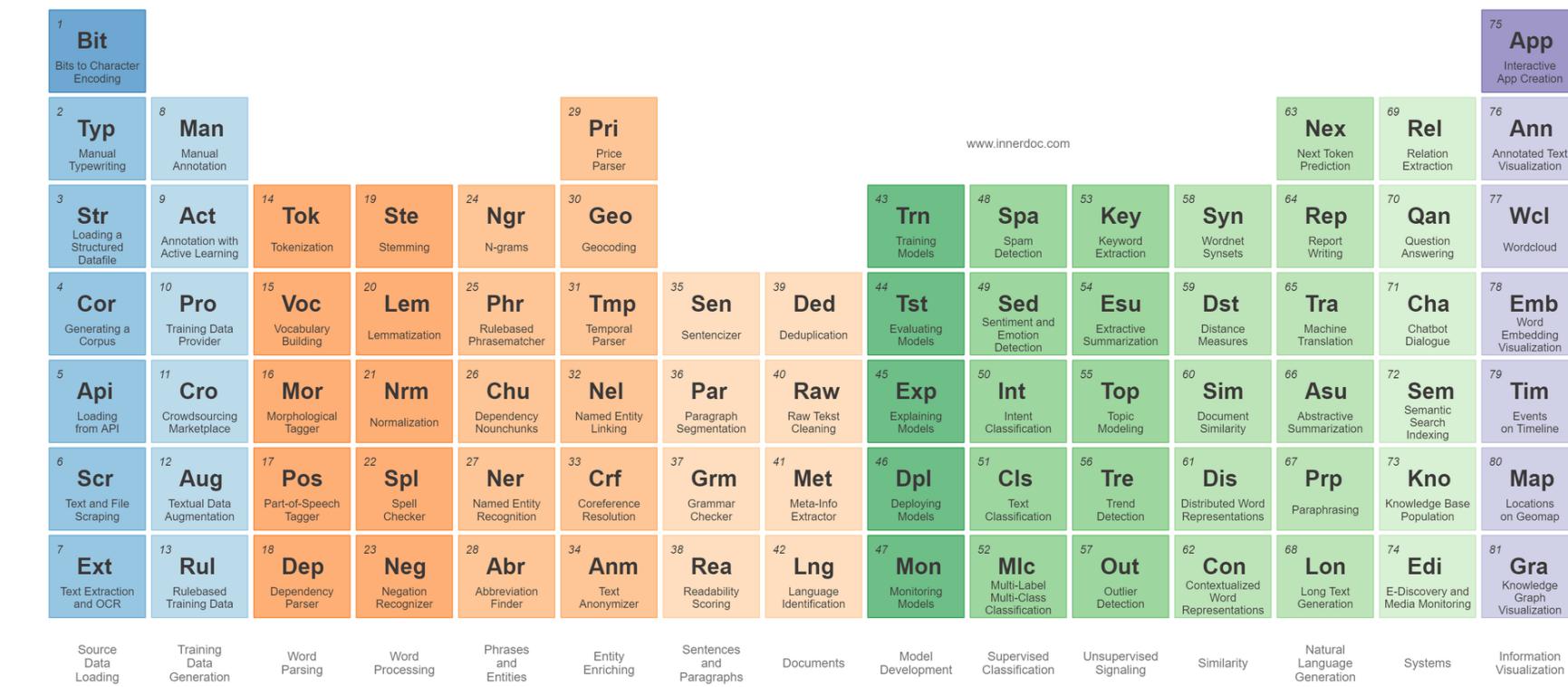
# 7. Processament del llenguatge natural

Models d'intel·ligència artificial



# Processament del llenguatge natural

- **Definició:** camp de la IA que tracta de la interacció entre els ordinadors i el llenguatge humà.
- Se centra en la **comprensió** i **generació** de llenguatge humà.
- Un dels camps més actius i complexos de la IA.



# Aplicacions

- Traducció automàtica.
- Reconeixement de veu.
- Síntesi de veu.
- Generació i resum de text.
- Anàlisi de sentiments.
- Classificació de text.

# Introducció (I)

- Camp multidisciplinari que combina:
  - Lingüística.
  - Intel·ligència artificial.
  - Ciències cognitives.
  - Informàtica.
  - etc.

# Introducció (II)

- És un problema **difícil** perquè:
  - El llenguatge humà és **ambigu**.
  - El llenguatge humà és **ric**.
  - El llenguatge humà és **contextual**.
  - El llenguatge humà és **cultural**.
- Aquestes característiques fan que el llenguatge humà no sigui **formal** i, per tant, no es puga tractar amb les tècniques de la IA tradicional.

# El text com a dada

# Introducció

- El text és una font de dades **molt important**.
- Els humans **generem i consumim** text de forma **massiva**
- El saber com tractar el text és **crític** per a moltes aplicacions.

Anomenem text a una seqüència coherent de símbols que pot ser interpretada com a un conjunt de paraules, utilitzant les regles gramaticals i sintàctiques d'una llengua.

# Significat

- Qué entenem per **significat**?:
  - el significat d'una paraula és el concepte que representa.
  - el significat d'una frase és el concepte que representa la combinació de les paraules que la formen.
  - el significat d'un text és el concepte que representa la combinació de les frases que el formen.

$\text{significant}(\text{simbol}) \Leftrightarrow \text{significat}(\text{idea})$   
 $\text{arbre} \Leftrightarrow \{ \text{🌲}, \text{🌳}, \text{🌴}, \dots \}$

# Significat en els ordinadors

- Com poden coneixer els ordinadors el significat de les paraules, frases i textos?.
- Técniques classiques: bases de dades de sinònims i hyperònims.
  - WordNet: base de dades lèxica que relaciona paraules entre si.
  - Els sinònims permeten relacionar paraules amb el mateix significat.
    - Ex: roïn és un sinònim de dolent, malparit, miserable, etc.
  - Els hyperònims permeten relacionar paraules amb significats més generals.
    - Exemple: carnivor, vertebrat serien hyperònims de gat.

# Problemes de WordNet i semblants

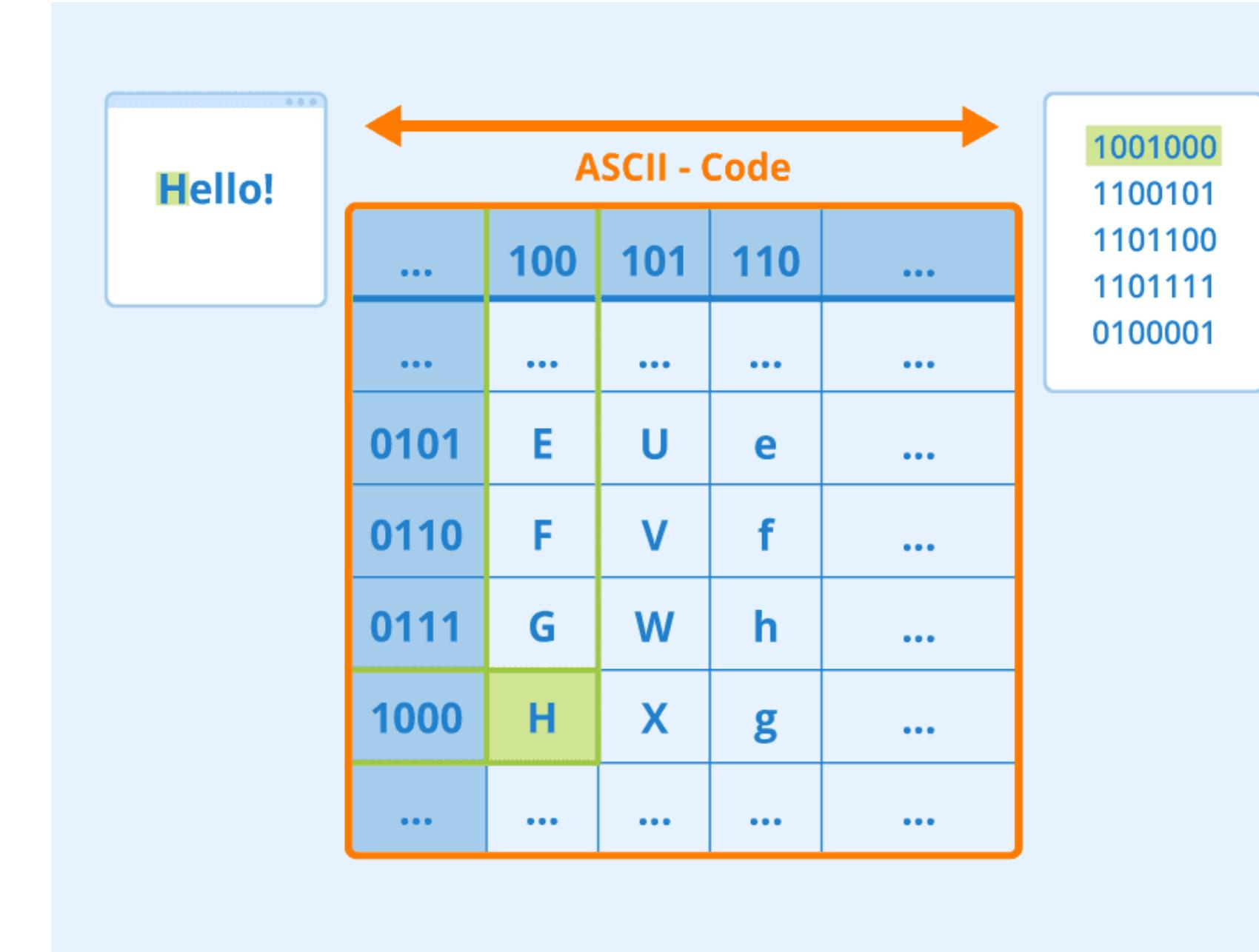
- Molt útil però sense matís semàntic.
  - Ex: gat i felí són sinònims, però no tenen el mateix significat.
- Tots els sinònims no seran útils en tots els contextos.
  - Ex: cabró és un sinònim de roïn, però no sempre es poden intercanviar.
- Actualitzar la base de dades és un procés **manual, costós** i subjectiu.
  - Ex: the shit és un sinònim de the best en anglès que no apareix en WordNet.
- Les solucions modernes es basen en les representacions del text.

# Representacions del text (I)

- Els ordinadors necessiten representar el text com a dades numèriques.
- Quina unitat de text triem per a representar el text i com ho fem?.
- Aquestes decisions determinaran la **complexitat** del model i la **qualitat** del resultat.
- Els models de representació del text són **molt importants** en el processament del llenguatge natural.
- A continuació veurem algunes de les tècniques més utilitzades.

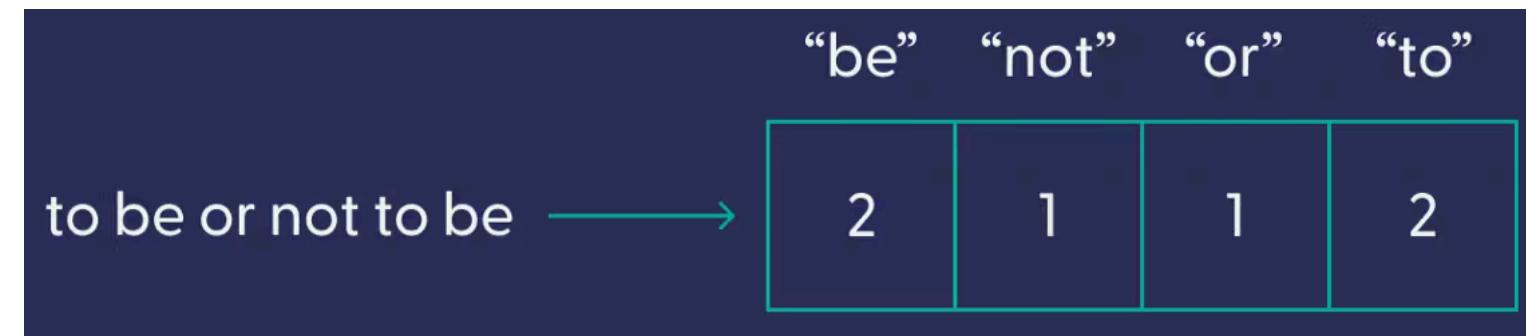
# Representació de caràcters

- Cada caràcter es representa per un número.
  - El diccionari serà molt petit (en el cas de l'ASCII, 128 caràcters).
  - El model ha de ser molt complex, ja que ha d'aprendre a combinar els caràcters per a formar paraules.
  - Exemple: "AND" es pot representar com a [65, 78, 68]



# Representació de paraules

- Cada paraula és representada per un número.
  - El diccionari serà molt gran, ja que cada paraula serà una entrada en el diccionari.
  - El model serà més senzill, ja que les paraules són unitats semàntiques.
  - Exemple: "BE" es pot representar com a 2.



# Representació de subparaules

- Cada subparaula és representada per un número.
  - El diccionari serà més petit que el de paraules, ja que les subparaules són unitats semàntiques.
  - El model serà més senzill, ja que les subparaules són unitats semàntiques.
  - Permet representar paraules rares i que no estan en el vocabulari.
  - Útil per a llengües amb moltes paraules compostes i derivades.

# Tokens i tokenització (I)

- Independentment de l'enfocament triat, el text ha de ser **dividit** en **tokens**.
  - Ex: "New York in winter" → [ "New", "York", "in", "winter" ]
- **Token**: unitat mínima de text que pot ser considerada com a una unitat semàntica.
  - Pot ser una paraula, una subparaula, un signe de puntuació, un número, etc; qualsevol unitat que es triï per a ser tractada com a unitat mínima.
- **Tokenització**: procés de dividir un text en tokens i que facilita el tractament i comprensió del text.
  - És un procés **no trivial**. Depèn de la llengua i del domini.
  - Exemple: "New York" és un token o dos?

# Tokens i tokenització (II)

- Qüestions a tindre en compte:
  - **Puntuació**: es considera un token o no?. Pot variar la interpretació del text.
  - **Majuscules/Minúscules**: es consideren tokens diferents o no?.
  - **Stopwords**: paraules que no aporten informació al text (articles, preposicions, etc.).
  - **Idioma i domini**: el procés de tokenització depèn de l'idioma i del domini del text.

# Tokens i tokenització (III)

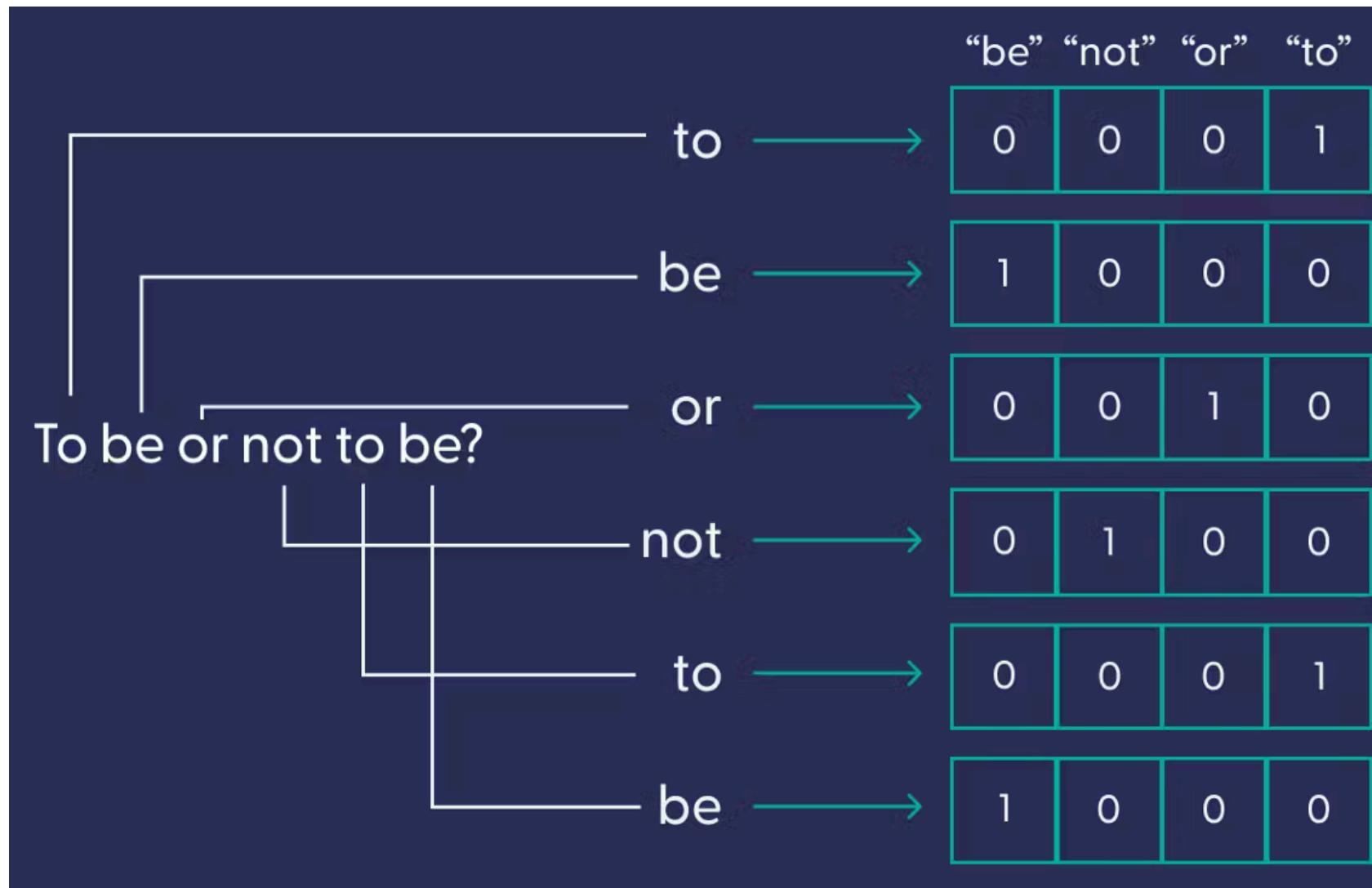
- N-grams: seqüències de n tokens consecutius.
- Algunes paraules tenen significat propi, però la seva combinació amb altres paraules també té un significat. Ex: "New York".
- Els n-grams permeten representar aquestes combinacions de paraules, augmentant el vocabulari amb les combinacions d'n-tokens que triem.
- Bigrams: seqüències de dos tokens consecutius, trigrams: seqüències de tres tokens consecutius, etc.
- Problema: augmenta molt el vocabulari i la complexitat del model.

# Representació de textos

- Fins ara hem vist com representar paraules.
- Solament així podrem veure les relacions entre les paraules que el formen.
- Algunes de les tècniques utilitzades són:
  - **One-hot encoding**: cada token → una dimensió; valor → 1 si el token està i 0 si no.
  - **Bag of Words**: model basat en freqüències. El valor de cada cel·la son les aparicions del token en el document. Simple i encara molt utilitzat.
  - **TF-IDF**: model basat en freqüències i inversa de freqüències
  - **Word Embeddings**. Vectorització de paraules amb *Word2Vec*, *FastText*, etc

# One-hot encoding

- El model **one-hot encoding** és un model basat en tokens.
- Els vectors generats són **dispersos** i **grans**, ja que cada token és una dimensió.
- Cada token és una dimensió i el valor de cada cel·la és 1 si el token està i 0 si no.
- Els vectors generats són **independents** de la semàntica.
- No facilita calcular la similitud entre paraules i textos.



## Raw Text

## Bag of Words (BoW)

- El model **BoW** és un model basat en freqüències.
- Es pot entendre com una suma dels vectors one-hot.
- Els vectors generats són **independents** de la semàntica.
- El nombre del token es pot entendre com a **ordre** i en molts casos no és així. Aquesta discrepància pot afectar a la qualitat del model.

it is a puppy and it  
is extremely cute

it is a puppy and it  
is extremely cute

Bag-of-words vector	
it	2
they	0
puppy	1
and	1
cat	0
aardvark	0
cute	1
extremely	1
...	...

# TF-IDF

- El model **TF-IDF** és un model basat en freqüències i inversa de freqüències.
- Semblant al model **BoW**, però té en compte la freqüència del token en el document i en el conjunt de documents.
- El valor de cada cel·la és el producte de la freqüència del token en el document i la inversa de la freqüència del token en el conjunt de documents.
- Dona més importància als tokens que apareixen en pocs documents. Raó: els tokens que apareixen en molts documents no soLEN aportar informació rellevant.

# Vectorització

- La **vectorització** és el procés de convertir un text en un vector numèric.
- Els vectors són més fàcils de manipular per a les màquines i de comparar.
- Algunes tècniques de vectorització (embeddings):
  - **NNLM**: model basat en xarxes neuronals. El nombre de dimensions és fixe.
  - **Word2Vec**: cada paraula és un vector en un espai semàntic. El nombre de dimensions és fixe.
  - **FastText**: creat per Facebook. Similar a Word2Vec, però permet representar paraules rares i que no estan en el vocabulari.
  - **GloVe**: model basat en NNLM i Word2Vec.

# Word2Vec

- Es basa en la idea que les paraules amb significats similars apareixen en contextos similars.
  - "You shall know a word by the company it keeps" (J.R. Firth, 1957).
- Quan una paraula **p** apareix en un text, les paraules properes a p són el seu **context**.
- Els diferents contextos de p defineixen el significat de p.

...government debt problems turning into **banking** crises as happened in 2009...

...saying that Europe needs unified **banking** regulation to replace the hodgepodge...

...India has just given its **banking** system a shot in the arm...



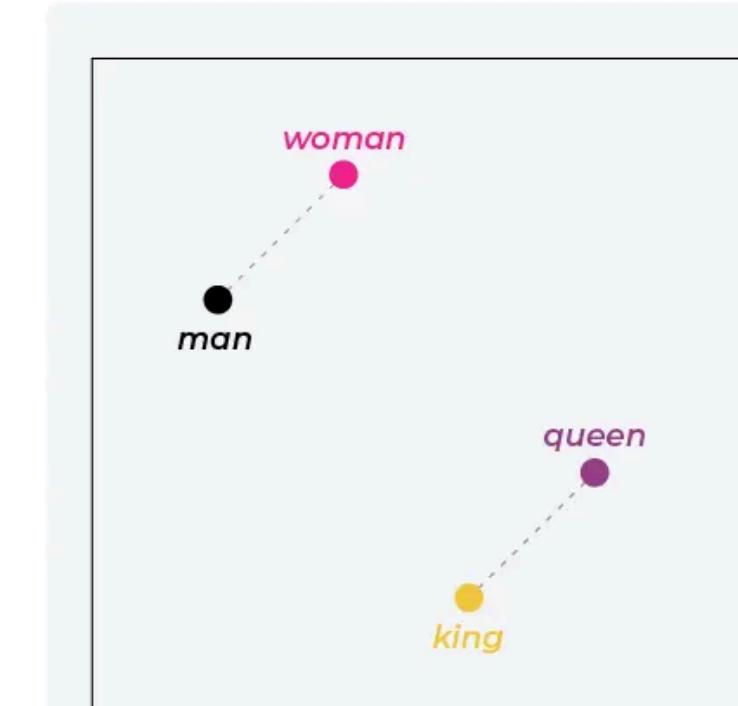
These **context words** will represent **banking**

# Word2Vec (II)

- Per cada paraula obtenim un vector **dens** i de **longitud fixa**.
- Cada dimensió del vector representa un **aspecte semàtic** de la paraula.
- Representen la seva **posició** en un espai semàtic n-dimensional.
- Els vectors de paraules amb contextos semblants estarán propers en l'espai semàtic.

Facilita calcular la similitud entre paraules.

	living being	feline	human	gender	royalty	verb	plural
man →	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
woman →	0.7	0.3	0.8	-0.7	0.1	-0.5	-0.4
king →	0.5	-0.4	0.7	0.8	0.9	-0.7	-0.6
queen →	0.8	-0.1	0.8	-0.9	0.8	-0.5	-0.9
word							



# Word2Vec (III)

- Els vectors de paraules també s'anomenen **embeddings** o **representacions de xarxa**.
- **FastText** és una variant de **Word2Vec** que utilitza subparaules.
  - Permet representar paraules rares i que no estan en el vocabulari.
  - Útil per a llengües amb moltes paraules compostes i derivades.
- Els embeddings generats poden ser utilitzats en una gran varietat de tasques, com pot ser la classificació de textos, anàlisi de sentiments, etc.
- Són models que necessiten un entrenament previ amb totes les paraules del vocabulari. A continuació veurem un exemple.

# Generació de *embeddings* amb Word2Vec i la llibreria Gensim

```
from gensim.models import Word2Vec  
  
sentences = [  
    ["Gavi", "company", "Pedri"],  
    ["Xavi", "entrena", "Barça"],  
    ["Gavi", "juga", "Barça"],  
    ["Gavi", "chuta"],  
    ["Kepa", "para"],  
    ["Xavi", "entrena"],  
]  
model = Word2Vec(sentences, min_count=1)
```

# Visualització dels embeddings

```
gavi = model.wv["Gavi"]
print(gavi)

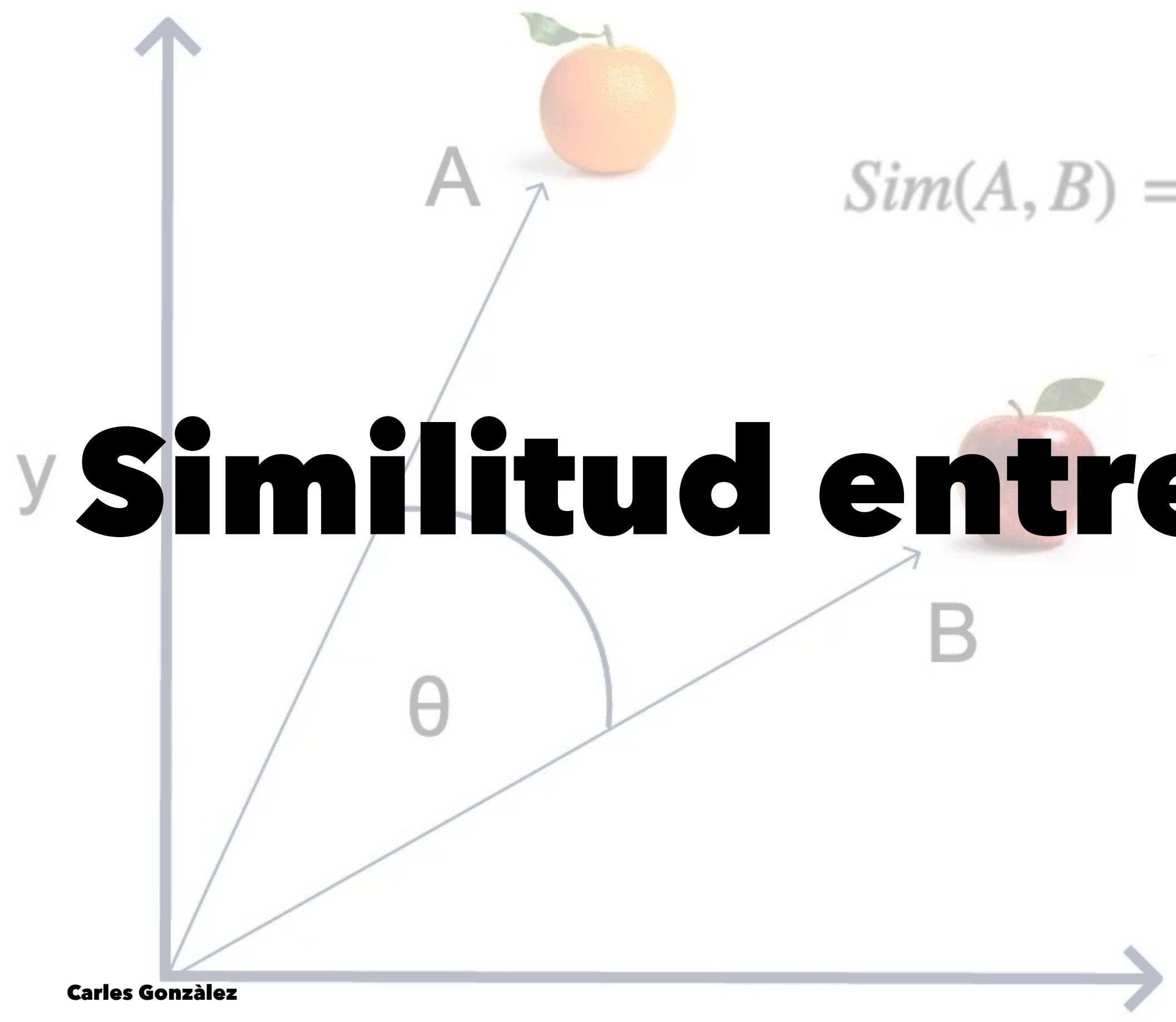
[
    -0.00536227  0.00236431  0.0510335   0.09009273
    -0.0930295   -0.07116809  0.06458873   0.08972988
    -0.05015428  -0.03763372
]
```

# Busquem paraules similars

```
model.wv.most_similar("Gavi")  
[( 'Barça' , 0.5436005592346191 ) ,  
  ( 'Pedri' , 0.3792896568775177 ) ,  
  ( 'entrena' , 0.3004249036312103 ) ,  
  ( 'Xavi' , 0.10494352877140045 ) ,  
  ( 'juga' , -0.1311161071062088 ) ,  
  ( 'chuta' , -0.1897382140159607 ) ,  
  ( 'para' , -0.22418655455112457 ) ,  
  ( 'Kepa' , -0.2726020812988281 ) ,  
  ( 'company' , -0.7287455797195435 )]
```

# Similitud del cosinus

- Els *embeddings* generats per Word2Vec són vectors de  $n$  dimensions.
- Per a calcular la similitud entre dos vectors s'utilitza la **similitud del cosinus**.
- El resultat és un valor entre -1 i 1.
  - -1: vectors oposats.
  - 0: vectors ortogonals.
  - 1: vectors iguals.
- Aquesta mesura és molt utilitzada en el processament del llenguatge natural.



$$Sim(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$

# Similitud entre textos

# Similitud entre textos

- La similitud entre textos és una mesura que indica com de semblants són dos textos.
- És una de les funcions més obvies del processament del llenguatge natural.
- El càlcul de la similitud entre textos, però, és una tasca **difícil**.
- Anem a veure algunes tècniques de les més utilitzades.

# Técniques per a calcular la similitud entre textos (I)

- **Basades en regles**: Es basen en regles predefinides; fàcils d'implementar i útils per a casos senzills.
  - **Distància de Levenshtein**: És el nombre mínim d'operacions per a transformar una cadena en una altra.
  - **Distància de Hamming**: És el nombre de posicions en les quals dues cadenes de la mateixa longitud difereixen.
  - **Recompte de paraules**: És el nombre comú de paraules entre dos textos.
  - **Distància de Jaccard**: És el nombre de paraules comunes entre dos textos dividit pel nombre total de paraules dels dos textos.

# Técniques per a calcular la similitud entre textos (II)

- **Basades en característiques sintàctiques:** Es basen en les característiques sintàctiques i gramaticals dels textos. Impliquen un procés de **parsejat** dels textos per analitzar la seva estructura sintàctica.
- **Basades en característiques semàntiques:** Es basen en les característiques semàntiques dels textos. Aquí models com Word2Vec són molt útils, al permetre representar el significat contextual de les paraules.
  - **Word Mover's Distance:** Mesura la distància entre dos textos com la distància entre els vectors de les paraules dels dos textos.
  - **Similitud del cosinus:** Utilitza el cosinus de l'angle entre ells.

# Técniques per a calcular la similitud entre textos (II)

- **Basades en l'aprenentatge automàtic**: Es basen en l'aprenentatge automàtic per a calcular la similitud entre textos.
- **BERT i GPT**: Models de llenguatge basats en xarxes neuronals que pot ser utilitzat per a calcular la similitud entre textos.
- **Universal Sentence Encoder**: Model específicament entrenat per al *transfer learning* (aprenentatge per a la transferència; utilitzar un model entrenat per a una tasca per a una altra).

# Utilitats de la similitud entre textos

- **Correcció ortogràfica:** Per a corregir una paraula es busca la paraula més semblant.
- **Classificació de textos:** Per a classificar un text es busca el text més semblant.
- **Agrupació de textos:** Útil per a agrupar textos similars en clusters.
- **Búsqueda de resposta:** Per a trobar la resposta a una pregunta es busquen texts semblants a la pregunta.

# Word Embeddings

- Les tècniques clàssiques d'NLP es basen en representacions no semàntiques com BoW o TF-IDF.
- Les modernes es basen en LLMs (Language Models) i Word Embeddings.
- Com ja hem vist, els *embeddings* generats per Word2Vec són vectors de  $n$  dimensions.
- Per a representar un text pot utilitzar-se la mitjana dels *embeddings* de les paraules que el formen.
- Els vectors generats són **denses**, de **longitud fixa** i amb **sentit semàtic**.
- Facilita calcular la similitud entre paraules i textos.

# LLMs (Language Models)

- Els **LLMs** són models complexos basats en xarxes neuronals recurrents i l'arquitectura **Transformer**.
- Poden aprendre la semàntica del text i generar els seus propis *embeddings* utilitzant el mecanisme d'**atenció**.
- Demostren un gran rendiment en moltes tasques, com pot ser el càcul de la similitud entre textos.
- Són complexos i necessiten un entrenament previ amb un gran volum de dades.

# Classificació de textos i anàlisi de sentiments

# Anàlisi de sentiments

- L'anàlisi de sentiments és un tipus de classificació i una de les tasques més utilitzades en el processament del llenguatge natural.
- L'objectiu és determinar l'actitud d'un autor respecte a un tema o producte.
- Es basa en la **polaritat** del text, que pot ser **positiva, negativa o neutra**.
- També poden buscar-se emocions concretes, com pot ser **alegria, tristesa, ira**, etc.

# Anàlisi subjectiva i objectiva

- L'anàlisi de sentiments pot ser **subjectiva** o **objectiva**.
- L'anàlisi subjectiva busca les **emocions i sentiments** de l'autor.
- L'anàlisi objectiva es basa en **fets i dades concretes**.
- Els dos tipus d'anàlisi són **complementaris** i poden utilitzar-se conjuntament.
- Ex: "*La pel·lícula té grans moments, però el final és molt trist*".
  - Anàlisi subjectiva: "*La pel·lícula és bona*".
  - Anàlisi objectiva: "*El final és trist*".

# Preprocessament del text

- El **tractament** de textos facilita obtenir bons resultats en NLP.
- Permet reduïr la **dimensionalitat** dels textos, eliminar el soroll i capturar la semàntica.
- Algunes de les tècniques més utilitzades són:
  - **Tokenització**: vist anteriorment.
  - **Normalització**: convertir el text a un format estàndard.
  - **Eliminació de stopwords**: eliminar paraules que no aporten informació.
  - **Stemming i lematització**: convertir les paraules a la seva forma base.
  - **Gestió de negacions i modalitats**: convertir a un format estàndard.

# Preprocessament: tokenització

- Com ja hem vist, la tokenització és el procés de dividir un text en tokens.
- Els tokens poden ser paraules, subparaules, signes de puntuació, etc.
- Facilita una anàlisi més profund del text i extreure característiques rellevants.
- Ex: "El Barça està en crisi" → ["El", "Barça", "està", "en", "crisi"].

# Preprocessament: normalització

- La **normalització** implica el·liminar els elements que no aporten informació.
  - Nombres, signes de puntuació, etc.
- També implica convertir el text a un format estàndard, passant a minúscules i llevant espais innecessaris, per exemple.
  - Ex: "El Barça està en crisi! 😠" → "el Barça està en crisi".
- La normalització facilita la comparació entre textos i la detecció de paraules clau.

# Preprocessament: eliminació d'**stopwords**

- Les **stopwords** són paraules que no aporten informació al text.
- Són paraules molt comunes en un idioma, com pot ser articles, preposicions, etc.
- Els textos després de processar-se amb stopwords són més fàcils de tractar i més ràpids de processar.
- Ex: "El Barça està en crisi" → ["Barça", "crisi"] .

# Preprocessament: stemming i lematització

- El **stemming** i la **lematització** són tècniques per a convertir les paraules a la seva forma base i facilitar l'agrupació de paraules relacionades.
- L'**stemming** és un procés heurístic basat en regles, mentre que la **lematització** és un procés basat en coneixements lingüístics.
  - Ex d'steming: "jugar", "jugaré", "jugarà" → "jug".
  - Ex de lematització: "jugar", "jugaré", "jugarà" → "jugar".
- El stemming és més ràpid, però la lematització és més precisa.

# Preprocessament: negacions i modalitats

- Les **negacions i modalitats** poden canviar el significat d'una frase.
- Ex: "*El Barça no està en crisi*", "*El Barça pot estar en crisi*", "*Deuries anar a l'estadi*".
- Els models de NLP no poden interpretar aquestes frases sense un tractament previ.
- Necessitem tècniques específiques, com la detecció de **dobles negacions** i la **reassignació de polaritat**.

# Enfocaments per a l'anàlisi de sentiments

- Com en totes les tasques de NLP, l'anàlisi de sentiments pot ser abordada amb diferents enfocaments.
- Els enfocaments més utilitzats són:
  - Basats en **regles**
  - Basats en l'**aprenentatge automàtic supervisat**
  - Basats en l'**aprenentatge automàtic no supervisat**

# Enfocament basat en regles

- Els enfocaments basats en regles són els més senzills i ràpids.
- També s'anomenen **lexicon-based**.
- Es basen llistes de paraules per a determinar la polaritat del text.
- De cada paraula es busca la seva polaritat en el llistat i es fa una suma..
  - "*El Barça està en crisi*". "crisi" → -1 - → **negatiu**.
  - "*Me fa il·lusió anar a l'estadi*". "il·lusió" → 1 → **positiu**.
  - "*El partit va acabar en empata*". "empat" → 0 → **neutre**.

# **Enfocament basat en l'AA supervisat**

- Els enfocaments basats en l'aprenentatge automàtic supervisat són alguns dels més utilitzats.
- Consisteixen en entrenar un model amb un conjunt de dades etiquetades amb la polaritat del text.
- El model identifica els patrons en el text que determinen la polaritat.
- Alguns dels models més utilitzats són: **Naive Bayes, Support Vector Machines, Random Forest, Xarxes neuronals**

# Enfocament basat en l'AA no supervisat

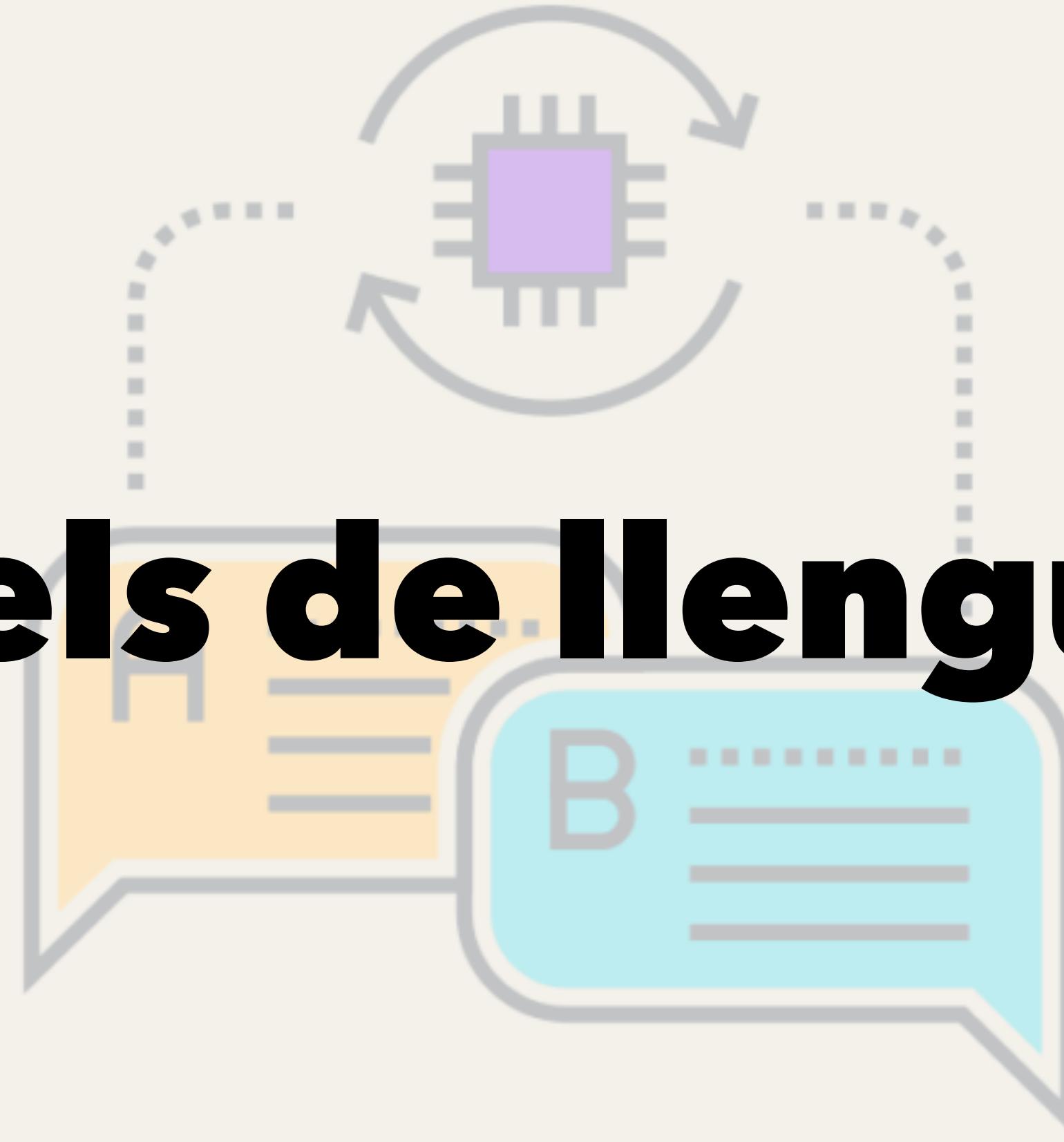
- Els enfocaments basats en l'aprenentatge es basen en identificar patrons en el text sense necessitar etiquetes predefinides.
- Es poden utilitzar tècniques com la **clusterització** per a agrupar els textos en clusters segons la seva polaritat.
- Una vegada agrupats els textos, es poden etiquetar manualment els clusters.
- Aquestes tècniques són útils per a detectar patrons en el text i per a agrupar textos semblants (segons la distància entre textos).

# Models per a l'anàlisi de sentiments

Alguns dels models més utilitzats per a l'anàlisi de sentiments són:

- **BOW** + clasificador: model basat en BoW i un classificador.
- **Embeddings** + clasificador: Word2Vec, FastText, etc.
- **VADER**: model basat en regles, molt utilitzat en anglès.
- **Transformers**: com ja hem vist, els transformers són models específics per a NLP molt potents. Un dels models més utilitzats és **BERT**, de quäl utilitzarem una implementació en la segona pràctica.

# Models de llenguatge



# Models de llenguatge

- Fins ara hem vist com representar el text i com acomplir tasques com l'anàlisi de sentiments.
- Hem comentat que els models de llenguatge són eines molt potentes que poden ser utilitzades en moltes tasques.
- En aquesta secció veurem què són els models de llenguatge i com funcionen.
- També veurem alguns dels més utilitzats i les seves aplicacions.

# Aplicacions dels models de llenguatge

- Els models de llenguatge són un dels camps més actius i complexos de la IA.
- Són la base de moltes aplicacions de NLP, com pot ser:
  - **Traducció automàtica**: traduir un text d'un idioma a un altre.
  - **Reconeixement de veu**: transcriure un text a partir d'un arxiu d'àudio.
  - **Síntesi de veu**: generar un arxiu d'àudio a partir d'un text.
  - **Generació i resum de text**: generar textos a partir d'una seqüència de paraules.
  - **Anàlisi de sentiments**: determinar la polaritat d'un text.
  - **Classificació de text**: classificar un text en una categoria.
  - **Generació de textos**: generar textos a partir d'un tema o un estil.

# Definició

- Un **model de llenguatge** assigna una probabilitat a una seqüència de paraules.
  - Per tant, permet predir la següent paraula d'una seqüència.
  - Ex: "El barça està en \_\_" → [{crisi: 0.8}, {forma: 0.1}, {casa: 0.1}]
- Es basen en la idea que les paraules d'una seqüència no són independents, sinó que depenen de les paraules anteriors.
- Permeten calcular la "**validesa**" d'una seqüència de paraules.
  - No és el mateix que la **correcció** d'una seqüència de paraules.
  - Intentem modelar el llenguatge humà, amb els seus **matisos i ambigüitats**.

# Història: Models basats en regles

- Els models de llenguatge són un dels camps més antics del processament del llenguatge natural.
- Es basen en regles gramaticals i lingüístiques per definir les característiques del llenguatge.
- Les regles estan definides per experts i són difícils de modificar.
- No són flexibles i no poden adaptar-se a nous contextos.
- Ex: Gramàtica de Chomsky, Gramàtica de Montague, etc.

# Història: Models estocàstics

- Els models basats en regles van ser substituïts per models basats en **estadístiques**, més flexibles i que poden modelar millor el llenguatge humà.
- Es basen en la idea que les paraules d'una seqüència no són independents, sinó que depenen de les paraules anteriors. Exemples:
  - **N-gram**: modela cada paraula en funció de les n paraules anteriors. (uni, bi, tri, etc).
  - **Skip-gram**: modela cada paraula en funció de les n paraules anteriors i posteriors.
  - **Syntax-based**: es basen en l'estructura sintàctica de les frases i no en la seva seqüència.

# Història: RNN

- Els models basats en estadístiques van ser substituïts per models basats en **xarxes neuronals**.
- Els primers models d'aquest tipus van ser els **RNN** (xarxes neuronals recurrents).
  - A diferència de les xarxes neuronals tradicionals, les RNN tenen **memòria**.
  - L'entrada d'una neurona pot anar determinada per la sortida d'ella mateixa.
  - Permeten processar seqüències de longitud variable.
  - Els models de llenguatge basats en RNN van ser els primers en obtenir resultats acceptables.

# Història: LLM

- Els models basats en RNN van ser substituïts per models basats en **transformers**.
- Els transformers són models basats en xarxes neuronals que utilitzen el mecanisme d'**atenció**.
  - Són més potents que les RNN i permeten obtenir resultats molt millors.
  - Són els models més utilitzats en l'actualitat.
  - Necessiten un entrenament previ amb un **gran** volum de dades (*corpus*)
  - Mostren la capacitat d'entendre el context, la semàntica i la sintàxis del text.

# Història: LLM (II)

- El mecanisme d'atenció és un mecanisme que permet a les xarxes neuronals aprendre a **centrar-se** en les parts importants de les seves entrades.
- És un mecanisme que imita el comportament humà.
- Podem entendre'l com una **capa** que s'afegeix a una xarxa neuronal.
- Els transformers són models basats en xarxes neuronals que utilitzen una variant del mecanisme d'atenció anomenada **self-attention**.

# Entrenament de models de llenguatge

- Els models de llenguatge necessiten un entrenament previ amb un **gran** volum de dades.
- Aquestes dades s'anomenen **corpus**.
- Els corpus són **molt grans** i difícils de generar.
- Poden ser **generats manualment** o **automàticament**.
- Varen en contingut: notícies, llibres, xats, etc.
- Poden incloure **etiquetes** per a entrenar models supervisats.

# Parts d'un LLM

- Els models de llenguatge basats en transformers són models molt complexos.
- Tenen dues parts principals:
  - **Encoder**: codifica el text d'entrada en un vector.
  - **Decoder**: decodifica el vector en un text de sortida.
  - Segons quines parts estiguem presents o no podran ser **bidireccionals** o **unidireccionals**.
  - Això determinarà també les tasques que poden realitzar.

# Parts d'un LLM (II)

- Encoder: codifica el text d'entrada en un vector.
- Els models encoder-only són **unidireccionals** i s'especialitzen en "entendre" el text d'entrada i, per tant, són útils per a tasques com la classificació de text.
- Solament necessiten el **encoder** per a realitzar la tasca perquè no necessiten generar un text de sortida.
- Utilitats: classificació de text, anàlisi de sentiments, etc.
- Ex: **BERT**, RoBERTa, ALBERT, ELECTRA, etc.

# Parts d'un LLM (III)

- Decoder: decodifica el vector en un text de sortida.
- Els models Decoder–only solament poden accedir a les paraules anteriors i, per tant, són útils per a tasques com la generació de text.
- Utilitats: generació de text, escritura creativa, etc.
- Ex: **GPT**, GPT-2, GPT-3, Mixtral, etc.

# Parts d'un LLM (IV)

- Encoder + Decoder: codifica el text d'entrada en un vector i decodifica el vector en un text de sortida.
- Els models Encoder–Decoder poden accedir a les paraules anteriors i posteriors i, per tant, són útils per a tasques com la traducció automàtica.
- Utilitats: traducció automàtica, resum de text, esquematització de text, etc.
- Ex: **T5**, BART, etc.

# Utilització dels LLM

- Els transformers són models molt complexos i necessiten un entrenament previ amb un gran volum de dades.
- Normalment s'utilitzen models ja entrenats i que poden ser utilitzats per a diferents tasques.
- Per a millorar el rendiment dels models entrenats es pot utilitzar el **fine-tuning**.
  - El fine-tuning consisteix en entrenar el model amb un conjunt de dades específic per a la tasca que volem realitzar.
  - Per a això, s'utilitza un conjunt de dades més petit que el corpus original.
  - En les pràctiques utilitzarem un model ja entrenat i li farem fine-tuning per a millorar el rendiment en les tasques que vulgarem realitzar.