

Escuela Latinoamericana de Aprendizaje Profundo

Curso de Lenguaje

Día 1

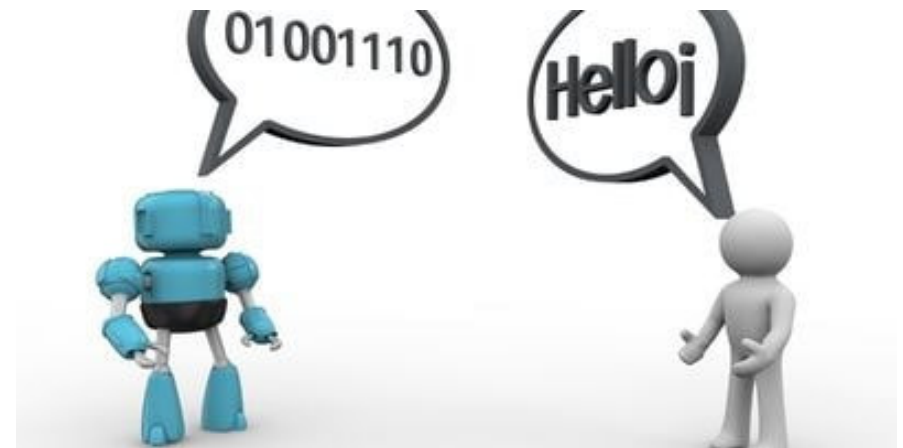
Alexandra González Eras

Lenguaje Natural



Las computadoras usan lenguajes formales para procesar el lenguaje natural con una máquina implica una traducción del primero al lenguaje de la última.

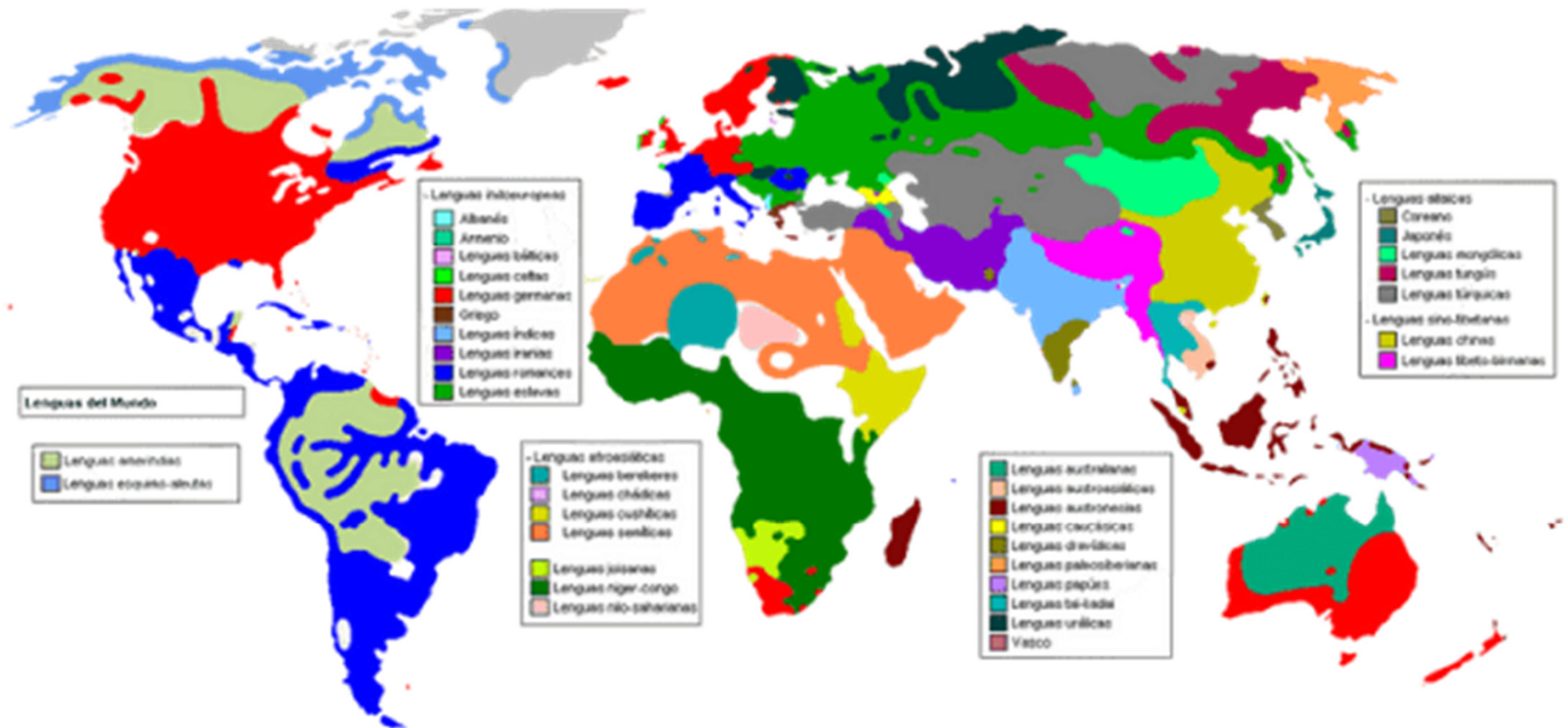
El lenguaje natural es el lenguaje humano, la lengua que un grupo o comunidad de gente ha establecido de manera espontánea para comunicarse entre ellos, transmitir pensamientos, ideas y conceptos así como para referenciar el mundo que les rodea.



El lenguaje es ambiguo

Componentes del lenguaje	Ambigüedad
Ambigüedad fonética y fonológica	<i>vaca / baca</i> <i>es conde / esconde</i>
Ambigüedad morfológica	<i>casa</i> → sustantivo o verbo <i>escribimos</i> → en ese momento o en el pasado
Ambigüedad sintáctica	<i>El otro día vi a Pedro pescando</i> ¿Pescaba yo o Pedro? ¿sujeto? ¿verbo? ¿predicado?
Ambigüedad semántica	Contexto: Lucas quiere ir a un cine Sinonimia: casa y morada Homonimia: banco y banco
Ambigüedad del discurso	« <i>Quiero eso</i> », donde el significado de «eso» depende de la situación
Ambigüedad pragmática	(« <i>Los manifestantes se alejaron de los violentos porque tenían miedo</i> » vs « <i>Los manifestantes se alejaron de los violentos porque gritaban mucho</i> »)

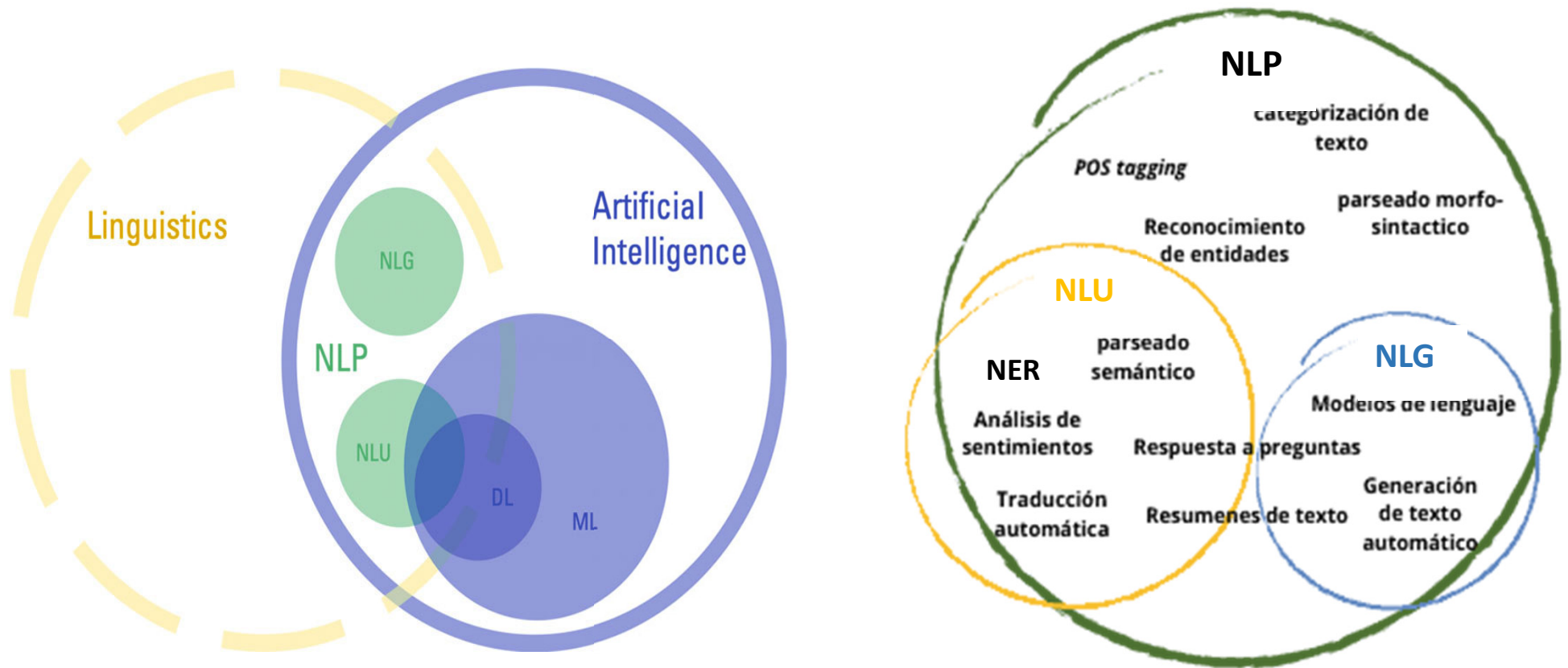
Diversidad Lingüística



A modo de resumen

- Por un lado, **el lenguaje natural es ambiguo, redundante, lleno de metáforas y de frases con segundos significados**, frente a las máquinas que son literales y concisas.
- **Hay muchos lenguajes y variedades del lenguaje**; y recursos solo para determinados idiomas.
- El **procesamiento de lenguaje natural** es un problema complejo y por tanto **requiere de la descomposición en tareas** que se tratan de **manera independiente** para luego integrar en sistemas más generales.

Procesamiento de Lenguaje Natural



NLP: Natural Language Processing

NLG: Natural Language Generation

NLU: Natural Language Understanding

Casos del PLN



Corpus

Un corpus es una gran colección de texto y, en el sentido del Machine Learning, un corpus puede considerarse como los datos de entrada de su modelo. El corpus contiene el texto que desea que aprenda el modelo.

Es común dividir un corpus grande en conjuntos de entrenamiento y prueba, utilizando la mayor parte del corpus para entrenar el modelo (80%) y una parte invisible del corpus para probar el modelo (20%)

IMDB: 50000 reseñas

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

<https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews>

Corpus: Representatividad y equilibrio

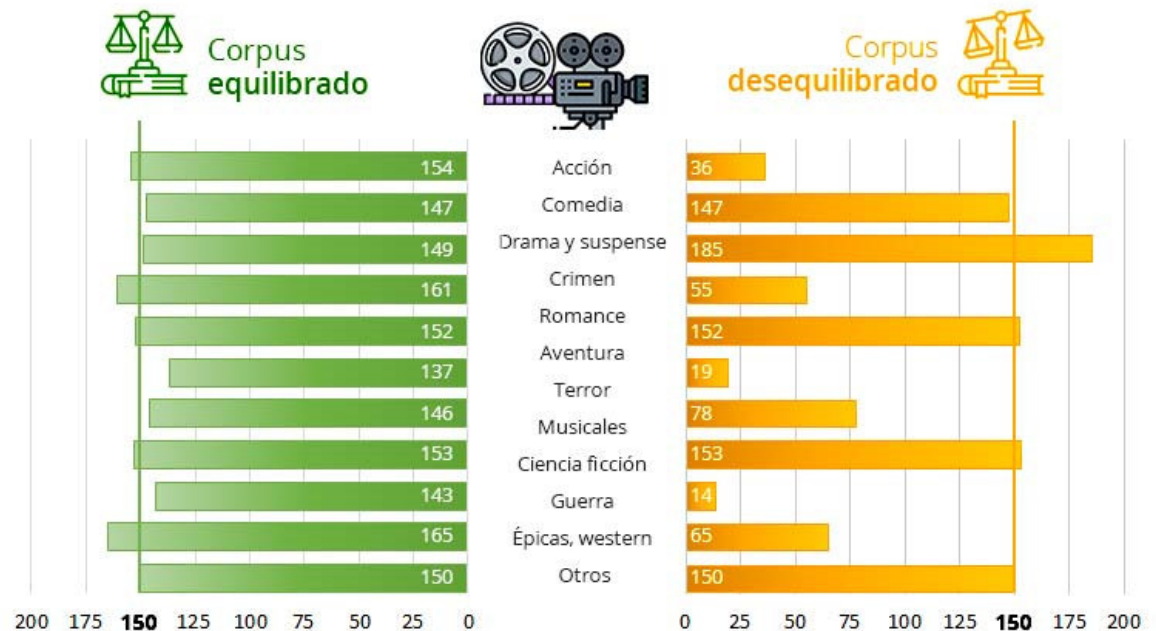
Leech (1991: 27) El nivel de representatividad de un corpus es alto cuando los descubrimientos o resultados obtenidos a partir de dicho corpus pueden generalizarse a otros datos que pertenecen al mismo dominio

Biber (1993: 43) explica que la representatividad de un corpus viene marcada por el grado en que una muestra incluye a toda la **gama de variabilidad de una población**.

la **representatividad de un corpus** viene determinada por el **grado de especificidad** del propio corpus

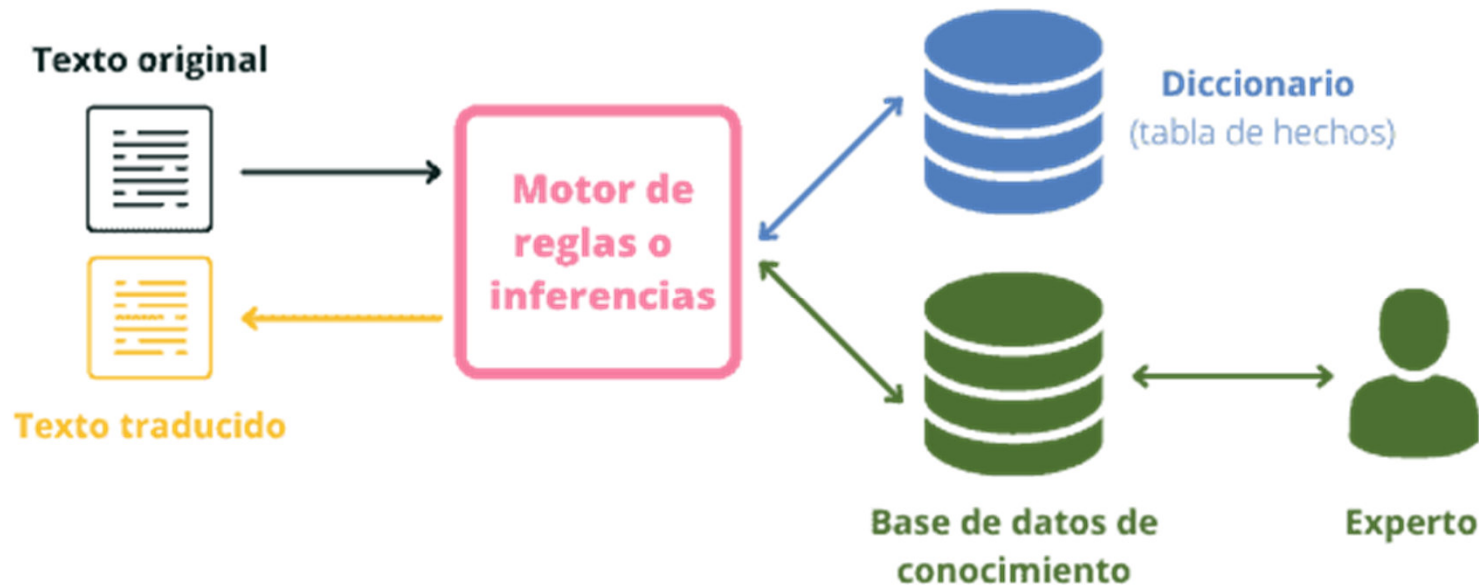
El equilibrio de un corpus proporción entre la cantidad de categorías o etiquetas asociadas al corpus y la cantidad de datos pertenecientes a cada una de ellas.

- Representación de corpus sobre géneros de cine -



Sistemas de reglas, expertos o basados en conocimiento:

Son sistemas donde **el lenguaje natural es traducido mediante una serie de reglas codificadas manualmente.**



PLN: Enfoque Estadístico

En estos sistemas hay **dos fases**: una de **entrenamiento**, donde la máquina aprende y **otra de ejecución** donde la máquina realiza la tarea.

Fase entrenamiento automático



Fase de ejecución



Procesamiento de texto



Normalización

```
def preprocess_text(sen):  
    # remueve tags html  
    sentence = remove_tags(sen)  
  
    # remueve puntuaciones y números  
    sentence = re.sub('[^a-zA-Z]', ' ', sentence)  
  
    # remueve caracteres individuales  
    #sentence = re.sub(r"s+[a-zA-Z]s+", ' ', sentence)  
  
    # remueve múltiples espacios  
    #sentence = re.sub(r's+', ' ', sentence)  
  
    return sentence
```

Mueve puntuaciones

Caracteres Individuales

Múltiples espacios

```
# para operar con expresiones regulares  
TAG_RE = re.compile(r'<[^\>]+>')  
  
#reemplaza cualquier cosa entre abrir y cerrar <>  
def remove_tags(text):  
    return TAG_RE.sub('', text)
```

Etiquetas HTML

procesa previamente las reseñas y se almacenan en un

```
X = []  
sentences = list(movie_reviews['review'])  
for sen in sentences:  
    X.append(preprocess_text(sen))
```

Estructuración

FreeLing 4.2 - An Open-Source Suite of Language Analyzers

Enjoy the FreeLing!

Write your sentences

Proclaiming that the main, most dangerous part of the war is already in the past; President Petro O. Poroshenko of Ukraine said on Thursday that his office was in constant communication with President Vladimir V. Putin of Russia to sustain a truce in eastern Ukraine, and that Ukraine was now looking ahead to pursue membership in the European Union.

Speaking at a news conference in Kiev, the capital, Mr. Poroshenko said that he planned to meet with Mr. Putin within the next three weeks and expressed confidence that the cease-fire with pro-Russian rebels would hold.

At the same time, Mr. Poroshenko voiced several positions certain to irk the Kremlin, which has worked aggressively to prevent Ukraine from shifting politically and economically toward Europe.

Analysis options

- ☒ Number recognition
- ☒ Date/Time recognition
- ☒ Quantities, ratios, and percentages
- ☒ Named Entity Recognition
- ☒ Multiword detection
- ☐ Phonetic encoding
- ☒ No sense annotation
- ☐ WN sense annotation: All senses
- ☐ WN sense annotation: [UKB](#) disambiguation

Select language

Auto-detect ▼

Select output

PoS Tagging ▼

Submit

<https://nlp.lsi.upc.edu/freeling/demo/demo.php>

Bolsas de palabras

1. This pasta is very tasty and affordable.
2. This pasta is not tasty and is affordable.
3. This pasta is very very delicious.

- this pasta is very tasty and affordable.
- this pasta is not tasty and is affordable.
- this pasta is very very delicious.

["and", "affordable.", "delicious.", "is", "not", "pasta", "tasty", "this", "very"]

	and	affordable	delicious	is	not	pasta	tasty	this	very
this pasta is very tasty and affordable.	1	1	0	1	0	1	1	1	1
this pasta is not tasty and is affordable	1	1	0	2	1	1	1	1	0
this pasta is very very delicious.	0	0	1	1	0	1	0	1	2

s.no	and	affordable	delicious	is	not	pasta	tasty	this	very
1	1	1	0	1	0	1	1	1	1
2	1	1	0	2	1	1	1	1	0
3	0	0	1	1	0	1	0	1	2

Oraciones iniciales

Normalización (mayúsculas)

Tokenización (Corpus)

Corpus: contiene unidades de significado textual (palabras, frases, párrafos hasta conjuntos de documentos)

Vectores de frecuencias

Sparse matrix

Código Bolsas de palabras

El método de la bolsa de palabras se usa principalmente en tareas de modelado de lenguaje y clasificación de texto. En este método, representaremos oraciones en vectores con la frecuencia de las palabras que aparecen en esas oraciones. En este enfoque realizamos dos operaciones.

Tokenización y Creación de Vectores

```
#Creating frequency distribution of words using nltk
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import CountVectorizer
import nltk
nltk.download('punkt')

text = """Achievers are not afraid of challenges, rather they relish them, thrive in them, use them. Challenges makes is stronger.
Challenges makes us uncomfortable. If you get comfortable with uncomfort then you will grow. Challenge the challenge """

#tokenize the sentences from the text corpus
tokenized_text=sent_tokenize(text)
#using CountVectorizer and removing stopwords in english language
cv1= CountVectorizer(lowercase=True,stop_words='english')
#fitting the tonized senetnecs to the countve torizer
text_counts=cv1.fit_transform(tokenized_text)
# printing the vocabulary and the frequency d strribution pf vocabulary in tokinzed sentences

print(cv1.vocabulary_)
print(text_counts.toarray())
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
{'achievers': 0, 'afraid': 1, 'challenges': 3, 'relish': 7, 'thrive': 9, 'use': 12, 'makes': 6, 'stronger': 8, 'uncomfortable': 11}
[[1 1 0 1 0 0 0 1 0 1 0 0 1]
 [0 0 0 1 0 0 1 0 1 0 0 0 0]
 [0 0 0 1 0 0 1 0 0 0 0 1 0]
 [0 0 0 0 1 1 0 0 0 0 1 0 0]
 [0 0 2 0 0 0 0 0 0 0 0 0 0]]
```

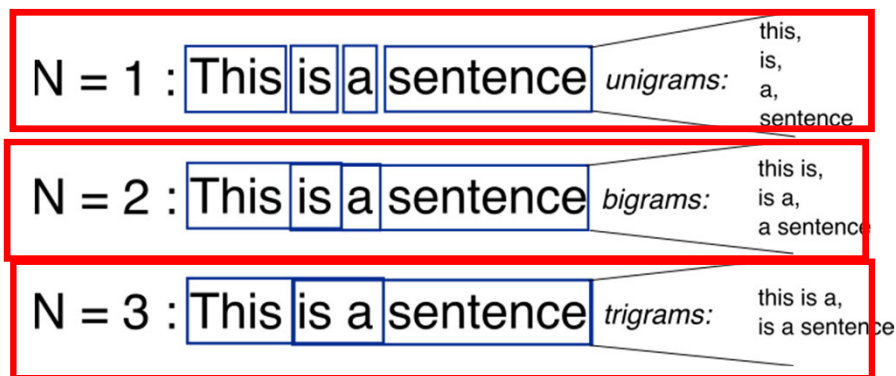
**Importa
librerías**

Tokenización

**Creación de
vectores**

N-grams

Definen una distribución de probabilidad sobre unidades de significado (palabras, términos), en función de la ocurrencia de las mismas en los textos. Un modelo n-gram usa las n-1 palabras anteriores para predecir la siguiente. Para ello, se requiere de la estimación a priori de una secuencia de palabras



Probabilidad de una sola palabra;

Palabra es estadísticamente dependiente de la palabra temporalmente anterior,

Palabra es estadísticamente dependiente de las dos palabras temporalmente anteriores

Extracting N-grams from the Text Data

```
▶ countvect = CountVectorizer(ngram_range = (2,2), )  
x_counts = countvect.fit(train_set.Message)  
  
# preparing for training set  
x_train_df = countvect.transform(train_set.Message)  
  
# preparing for test set  
x_test_df = countvect.transform(test_set.Message)
```

One-hot encoding

Representa texto como una serie de unos y ceros.

$V = \{\text{zebra, horse, school, summer}\}$

$V(\text{zebra}) = [1,0,0,0]$

$V(\text{horse}) = [0,1,0,0]$

$V(\text{horse}) = [0,0,1,0]$

$V(\text{horse}) = [0,0,0,1]$

```
from numpy import argmax
# define input string
data = 'hello world'
print(data)
print()
# define universe of possible input values
alphabet = 'abcdefghijklmnopqrstuvwxyz '
# enumerate the alphabet
print(list(enumerate(alphabet)))
print()
# define a mapping of chars to integers
char_to_int = dict((c, i) for i, c in enumerate(alphabet))
int_to_char = dict((i, c) for i, c in enumerate(alphabet))
# integer encode input data
integer_encoded = [char_to_int[char] for char in data]
print(integer_encoded)
print()
```

Define el universo de posibles valores de entrada

Define el universo de posibles valores de entrada

One-hot encoding

Representa texto como una serie de unos y ceros.

```
# one hot encode
onehot_encoded = list()
for value in integer_encoded:
    letter = [0 for _ in range(len(alphabet))]
    letter[value] = 1
    onehot_encoded.append(letter)
print(onehot_encoded)
# invert encoding
inverted = int_to_char[argmax(onehot_encoded[0])]
print(inverted)
```

hello world

```
[('0', 'a'), (1, 'b'), (2, 'c'), (3, 'd'), (4, 'e'), (5, 'f'), (6, 'g'), (7, 'h'), (8, 'i'), (9, 'j'), (10, 'k'), (11, 'l'), (12, 'm'), (13, 'n')]
```

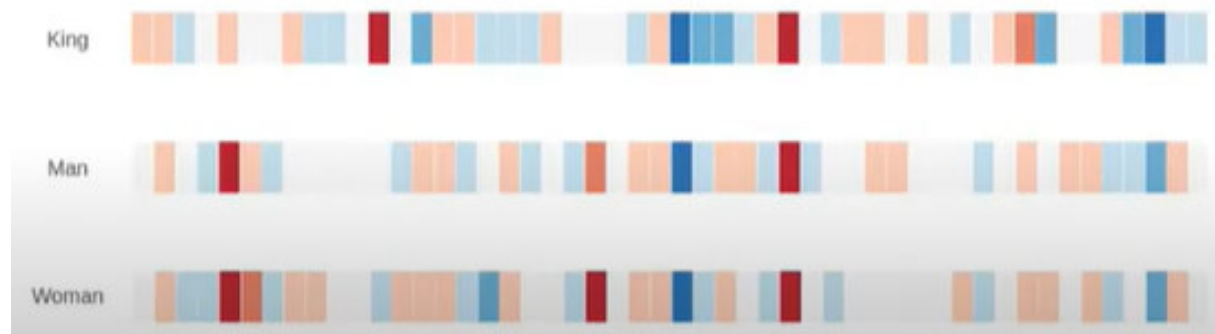
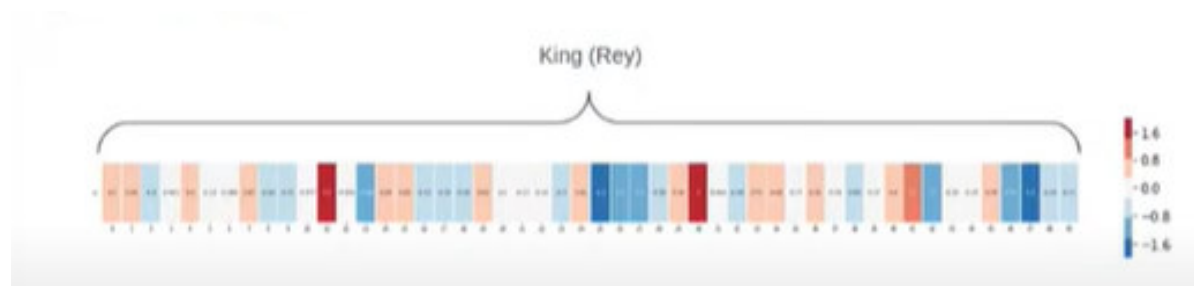
[7, 4, 11, 11, 14, 26, 22, 14, 17, 11, 3]

$$h = [[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],$$

Enumeración del vocabulario

Representación en un one-hot vector

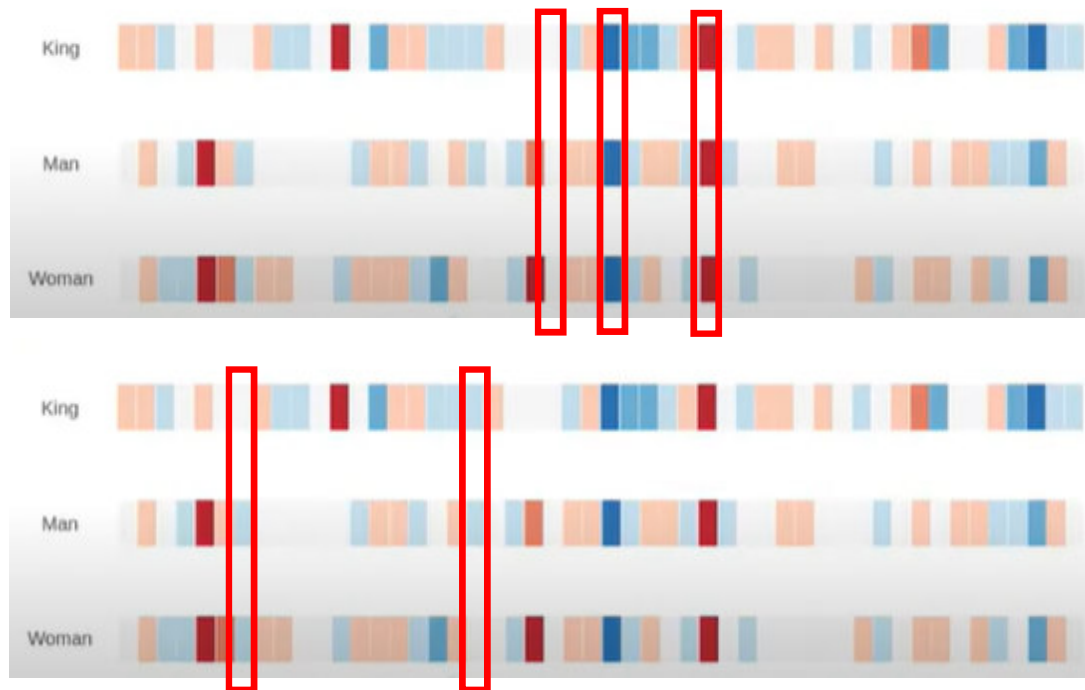
Modelo de lenguaje Word2Vec



Técnica basada en Aprendizaje Automático que aprende asociaciones de palabras a partir de corpus de texto.

Cada una de las palabras es codificada como un vector de números en un espacio dimensional, que puede ser emparejado con otros vectores que aparecen en contextos similares

Modelo de lenguaje Word2vec



Existen atributos similares que permiten definir similitudes entre las palabras

Existen atributos similares que permiten definir diferencias entre las palabras

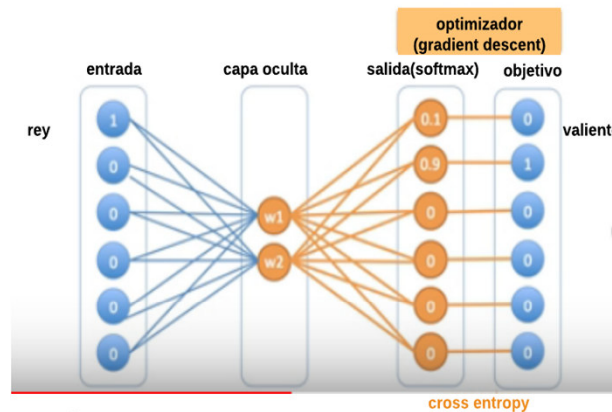
Modelo de lenguaje Word2Vec

2 Word2Vec ventana = 2

palabra	palabra one hot vector	vecino	vecino one hot vector
rey	[1,0,0,0,0,0]	valiente	[0,1,0,0,0,0]
rey	[1,0,0,0,0,0]	hombre	[0,0,1,0,0,0]
valiente	[0,1,0,0,0,0]	rey	[1,0,0,0,0,0]
valiente	[0,1,0,0,0,0]	hombre	[0,0,1,0,0,0]
hombre	[0,0,1,0,0,0]	rey	[1,0,0,0,0,0]
hombre	[0,0,1,0,0,0]	valiente	[0,1,0,0,0,0]
reina	[0,0,0,1,0,0]	hermosa	[0,0,0,0,1,0]
reina	[0,0,0,1,0,0]	mujer	[0,0,0,0,0,1]
hermosa	[0,0,0,0,1,0]	reina	[0,0,0,1,0,0]
hermosa	[0,0,0,0,1,0]	mujer	[0,0,0,0,0,1]
mujer	[0,0,0,0,0,1]	reina	[0,0,0,1,0,0]
mujer	[0,0,0,0,0,1]	hermosa	[0,0,0,0,1,0]

1 Texto de entrenamiento
 rey valiente hombre
 reina hermosa mujer

3 Entrenamiento



4 Modelo de representación vectorial

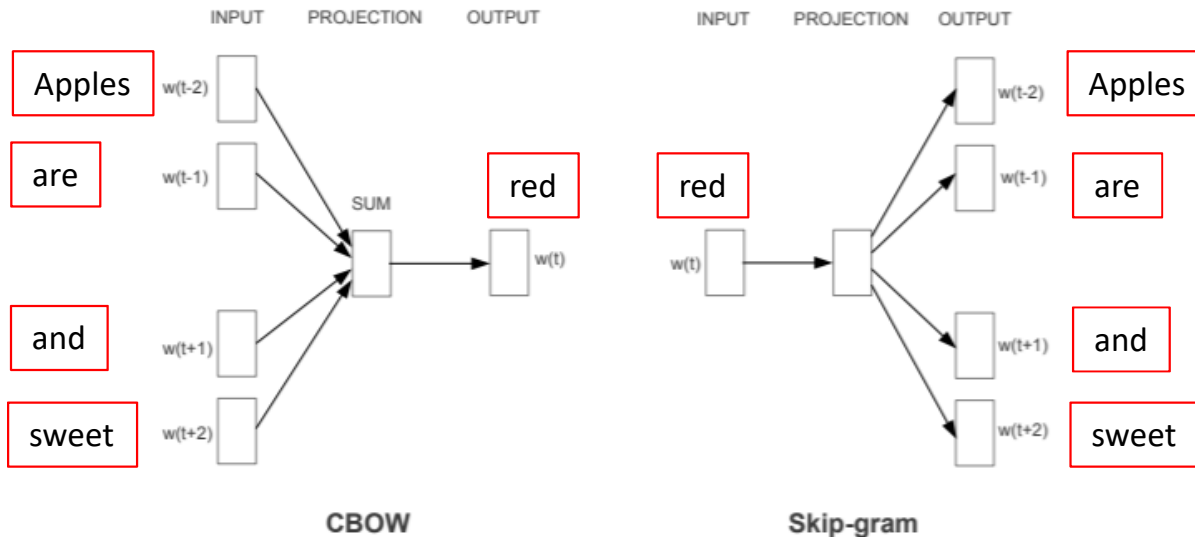
palabra	vector	word embedding
rey	[1,0,0,0,0,0]	[1,1]
hombre	[0,0,1,0,0,0]	[1,3]
reina	[0,0,0,1,0,0]	[5,5]
mujer	[0,0,0,0,0,1]	[5,7]



Se considera la vecindad de las palabras
 “rey” es vecino de “valiente” y de “hombre”
 “valiente” es vecino de “rey” y de “hombre”
 “hombre” es vecino de “valiente” y de “rey”)

Cross entropy: la entropía cruzada entre dos distribuciones de probabilidad p y q sobre el mismo conjunto de eventos, mide el número promedio de bits necesarios para identificar un evento que ocurre en el conjunto.

Modelos de lenguaje Word2vec



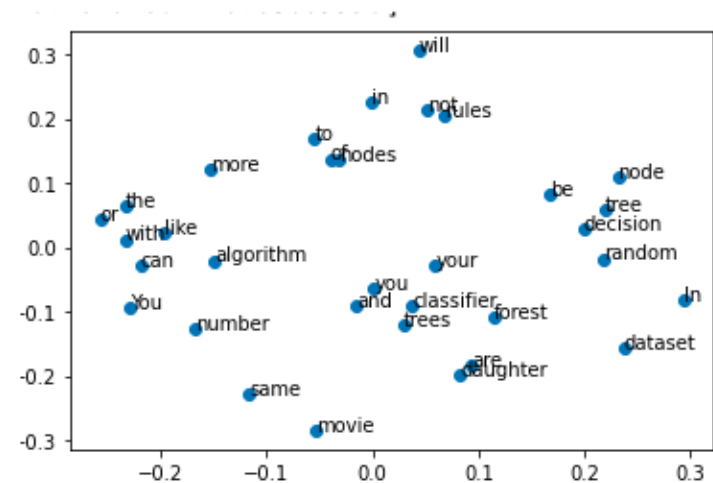
El modelo **CBOW** obtiene una representación (o predicción de una palabra central a partir de las palabras que se sitúan alrededor de ella (contexto)

El modelo **skip-gram** obtiene el contexto, representación (o predicción) de una palabra central a partir de las palabras que se sitúan alrededor de ella (contexto)

Word2vec Cbow

```
1 # Predict center word using context words within window size
2 # import word2vec class from gensim
3 from gensim.models import Word2Vec
4 # apply word2vec to sentences
5 model_cbow = Word2Vec(sentences=word_tokenizer,
6                        size=2, # number of dimensions default 100
7                        window=2, # window size default 5
8                        min_count=2, # minimum count of words to be consider default 5
9                        workers=1, # workers default 3
10                       sg=0 # method 0 for CBOW and 1 for skip gram
11                       )
12
13 # total vocabulary words for word 2 vec model
14 words_cbow = list(model_cbow.wv.vocab)
15 print(f"Total number of words {len(words_cbow)}")
16 # we can get word embedding value for a particular word
17 model_cbow.wv.__getitem__('tree')
```

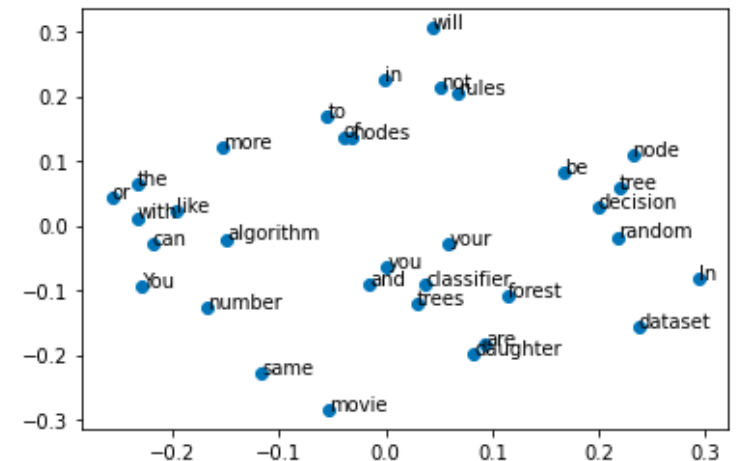
```
1 X = model_cbow.wv.__getitem__(model_cbow.wv.vocab)
2 # create 2D model using PCA
3 pca_model = PCA(n_components=2)
4 result = pca_model.fit_transform(X)
5
6 # visualize pca model using matplotlib
7 plt.scatter(result[:,0], result[:,1])
8 words_cbow = list(model_cbow.wv.vocab)
9 for i, word in enumerate(words_cbow[:100]):
10     plt.annotate(s=word,xy=(result[i,0], result[i,1]))
11 plt.show()
```



Word2vec skip-gram

```
1 # import word2vec class from gensim
2 from gensim.models import Word2Vec
3 # apply word2vec to sentences
4 model_skip = Word2Vec(sentences=word_tokenizer,
5                        size=2, # number of dimensions default 100
6                        window=2, # window size default 5
7                        min_count=4, # minimum count of words to be consider default 5
8                        workers=1, # workers default 3
9                        sg=1 # method 0 for CBOW and 1 for skip gram
10                       )
11
12 # total vocabulary words for word 2 vec model
13 words = list(model_skip.wv.vocab)
14 print(f"Total number of words {len(words)}")
15
16 # we can get word embedding value for a particular word
17 wordembedding = model_skip.wv.__getitem__('tree')
18 print(wordembedding)
19
20
```

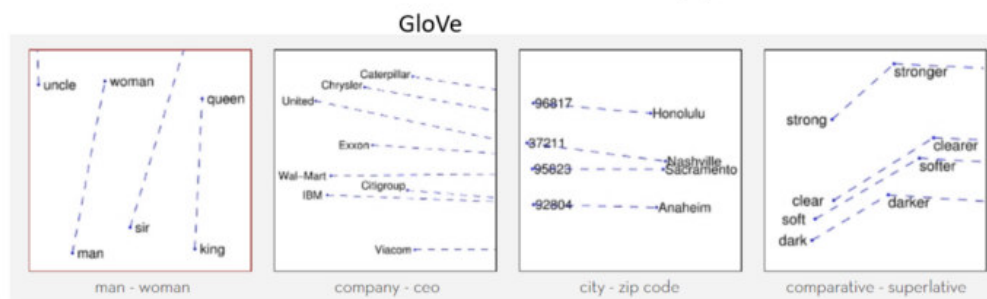
```
1 X = model_skip.wv.__getitem__(model_skip.wv.vocab)
2 # create 2D model using PCA
3 pca_model = PCA(n_components=2)
4 result = pca_model.fit_transform(X)
5
6 # visualize pca model using matplotlib
7 plt.scatter(result[:,0], result[:,1])
8 words_skip = list(model_skip.wv.vocab)
9 for i, word in enumerate(words_skip[:100]):
10     plt.annotate(s=word, xy=(result[i,0], result[i,1]))
11 plt.show()
```



Modelos de lenguaje: GloVe

GloVe es un algoritmo de aprendizaje no supervisado para obtener representaciones vectoriales de palabras.

El entrenamiento se realiza en estadísticas globales agregadas de co-ocurrencia palabra-palabra de un corpus, y las representaciones resultantes muestran subestructuras lineales interesantes del espacio vectorial de palabras.



```
# matriz de características con incrustaciones GloVe
from numpy import array
from numpy import asarray
from numpy import zeros

embeddings_dictionary = dict()
glove_file = open("./glove.6B.100d.txt")

for line in glove_file:
    records = line.split()
    word = records[0]
    vector_dimensions = asarray(records[1:], dtype="float32")
    embeddings_dictionary[word] = vector_dimensions
glove_file.close()
```

A modo de resumen

- Los **modelos basados en conocimiento** requieren del **experto y de recursos lingüísticos**, para validar los resultados de los análisis de Procesamiento de Lenguaje Natural
- Los modelos como **one-hot transforman el texto en valores numéricos** y representan las palabras en forma de **largos vectores de ceros y unos**. Son **fáciles de implementar**, pero **no incluyen el contexto de la palabra**
- Las **incrustaciones** permiten generar **estructuras multidimensionales** donde se logra la **representación de la palabra y su contexto**. Para ello, se apoyan en recursos como las **incrustaciones de Glove**, que **enriquece semánticamente** el contexto de la palabra