spaCy

Business Intelligence – Machine Learning – Big Data

1 . SPACY

INTRODUCCIÓN

spaCy es una librería de Python que permite construir aplicaciones de procesamiento de lenguaje natural (NLP). spaCy proporciona modelos preentrenados de diferentes lenguajes, lo cual junto a una sintaxis clara hace que sea ideal para principiantes en el campo de la NLP. Además, permite crear modelos nuevos o reentrenar los modelos que proporciona con datos propios para entrenar los modelos en campos específicos. spaCy está orientado al uso de producción, proporcionando un framework para construir aplicaciones completas que requieran de procesamiento de lenguaje natural, a diferencia de otras librerías como TensorFlow que permiten experimentar con diferentes arquitecturas de redes neuronales o implementar los últimos modelos desarrollados. Además, spaCy es eficiente en CPU pese a utilizar modelos de redes neuronales. La velocidad y precisión de los modelos de spaCy son de los mejores en el mercado.

Los modelos de spaCy de procesamiento de lenguaje natural permiten analizar un texto y extraer información tanto del texto como de las predicciones del modelo sobre su significado por el contexto. Son útiles por ejemplo para el análisis de artículos, de tweets, de páginas web... spaCy permite mucha personalización tanto en el entrenamiento de modelos como el añadir información al análisis del texto para crear aplicaciones a medida.

ESTRUCTURAS DE DATOS

spaCy tiene tres objetos principales para el análisis de un texto. El objeto más fundamental es Token. El texto procesado se divide en Tokens, que constituyen palabras o signos de puntuación. Cada Token de spaCy tiene ciertos atributos e información, por ejemplo: el texto del Token (la propia palabra o signo de puntuación), información del análisis según el contexto, como puede ser el tipo de palabra (nombre, adjetivo, verbo...), la dependencia sintáctica con otro Token... La colección de Tokens del texto constituye un Doc, mientras que un Span es un trozo de Doc.

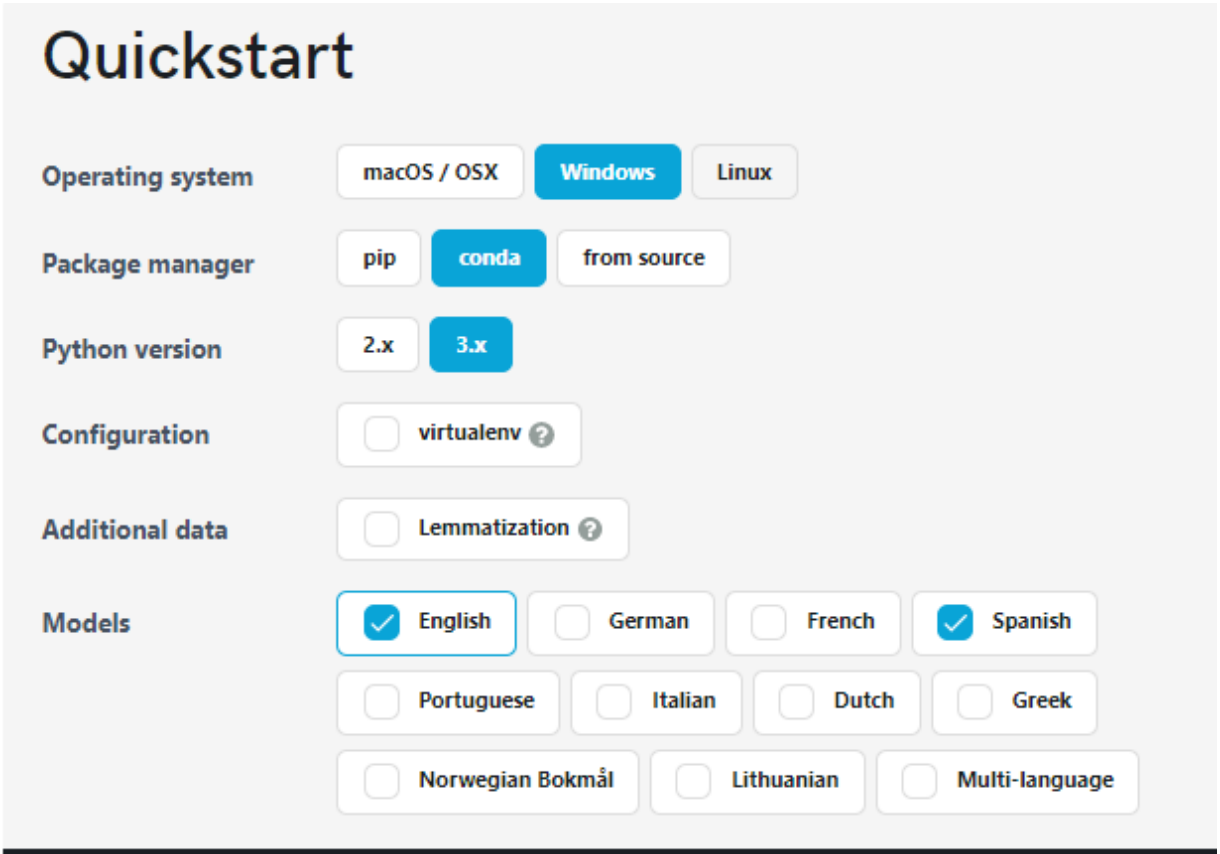
Cada objeto de spaCy tiene sus atributos que aportan mucha información sobre el análisis que es interesante saber, y spaCy además permite añadir nuevos atributos, propiedades o métodos personalizados.

En el siguiente apartado veremos una pequeña introducción a las características de la librería para poder ver el alcance y la potencia de desarrollo de aplicaciones con spaCy.

2 .EJEMPLOS DE USO

INSTALACIÓN

La instalación de spaCy puede hacerse mediante conda o pip. En la página web spacy.io/usage se encuentran instrucciones para el comando específico a ejecutar. Por ejemplo, para instalar spaCy por conda en Windows, justo con los modelos de inglés y español, la página indica los siguientes comandos:




```
$ conda install -c conda-forge spacy
$ python -m spacy download en_core_web_sm
$ python -m spacy download es_core_news_sm
```

IMPORTAR EL MODELO Y PRIMEROS USOS

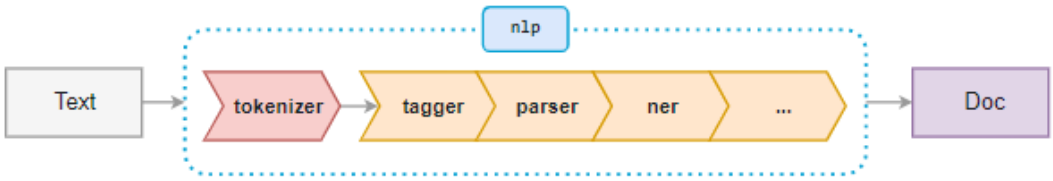
Una vez instalada la librería junto con algún modelo que proporciona spaCy, lo siguiente es importarlo en Python y utilizarlo. Esto se hace mediante el método load, el nombre del modelo se puede ver en la imagen anterior. En este caso, el modelo del idioma español es "es_core_news_sm". El "sm" final del nombre del modelo significa "small", modelo pequeño. Según el idioma spaCy proporciona modelos de distinto tamaño. Existen diferencias entre cada modelo, la más notable es que los modelos pequeños no tienen vectores de palabras, que se utilizan para calcular la similaridad.

```
[1]: import spacy

     nlp = spacy.load("es_core_news_sm")

     doc = nlp("Esto es una frase de ejemplo.")
     doc

[1]: Esto es una frase de ejemplo.
```

Al llamar al modelo en una cadena de texto, el modelo procesa el texto y genera un objeto Doc. El objeto Doc es una colección de objetos Token. Podemos acceder a ellos mediante un índice numérico, como haríamos en listas o cadenas. Además podemos utilizar índices para cortar el Doc y generar un Span.

```
[5]: type(doc)

[5]: spacy.tokens.doc.Doc

[6]: type(doc[2:4])

[6]: spacy.tokens.span.Span

[3]: type(doc[0])

[3]: spacy.tokens.token.Token
```

El procesamiento del texto se hace mediante el pipeline. El pipeline es un conjunto de funciones que se aplican a un objeto Doc, de manera secuencial (una a continuación de la otra). El pipeline por defecto de los modelos de spaCy se puede ver en la siguiente imagen.



NAME	COMPONENT	CREATES	DESCRIPTION
tokenizer	`Tokenizer`	`Doc`	Segment text into tokens.
tagger	`Tagger`	`Doc[i].tag`	Assign part-of-speech tags.
parser	`DependencyParser`	`Doc[i].head`, `Doc[i].dep`, `Doc.sents`, `Doc.noun_chunks`	Assign dependency labels.
ner	`EntityRecognizer`	`Doc.ents`, `Doc[i].ent_iob`, `Doc[i].ent_type`	Detect and label named entities.
textcat	`TextCategorizer`	`Doc.cats`	Assign document labels.
...	custom components	`Doc._.xxx`, `Token._.xxx`, `Span._.xxx`	Assign custom attributes, methods or properties.

Primero se ejecuta el tokenizer que genera los Token y con ello el Doc (colección de tokens). Después las siguientes componentes reciben el Doc y lo procesan, añadiendo información de manera sucesiva. Las componentes que vienen por defecto son el Tagger, el Parser y el NER.

Durante el tratamiento del texto mediante spaCy no se pierde información. Tanto los Token como los Doc guardan el texto original además de la información obtenida por el modelo. Para ahorrar memoria, spaCy almacena algunos campos codificados, para acceder al valor sin codificar se utiliza "_" al final del nombre del campo. Algunos ejemplos de campos de interés de un Token pueden ser:

```
[12]: token = doc[0]
      print('texto: ', token.text)
      print('índice: ', token.i)
      print('es número: ', token.like_num)
      print('es alfanumérico: ', token.is_alpha)
      print('es puntuación: ', token.is_punct)
      print('part of speech: ', token.pos_)
      print('dependencia: ', token.dep_)
```

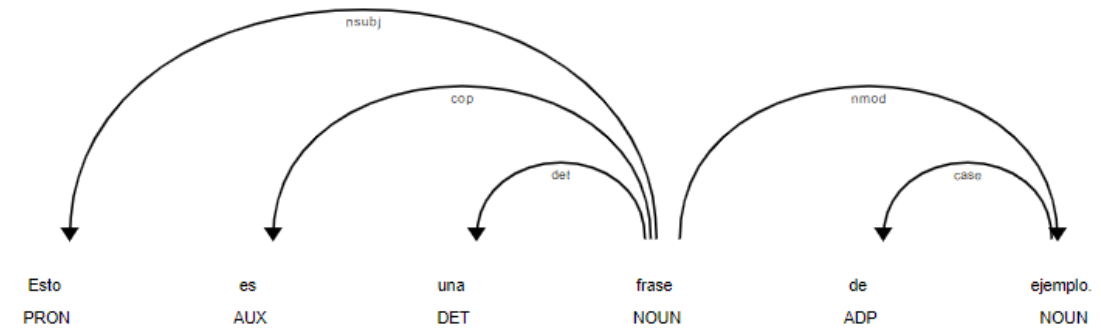
```
texto:  Esto
índice:  0
es número:  False
es alfanumérico:  True
es puntuación:  False
part of speech:  PRON
dependencia:  nsubj
```

En el caso de no saber qué significa alguna etiqueta o abreviatura, spaCy cuenta con la función "explain" que da más información respecto al significado de las etiquetas, por ejemplo:

```
[14]: spacy.explain(token.dep_)

[14]: 'nominal subject'
```

Además, spaCy también permite visualizar el análisis de dependencias sintácticas mediante displacy.

```
[16]: from spacy import displacy

      displacy.render(doc)
```



Podemos estructurar la información de los diferentes token.

```
[17]: import pandas as pd

df = pd.DataFrame([[token.text, token.lemma_, token.pos_, token.tag_,
                    token.dep_, token.shape_, token.is_alpha, token.is_stop] for token in doc])
df.columns = ['texto', 'lemma', 'pos', 'tag', 'dep', 'forma', 'alpha', 'stop']
df
```

[17]:

	texto	lemma	pos	tag	dep	forma	alpha	stop
0	Esto	Esto	PRON	PRON__Number=Sing|PronType=Dem	nsubj	Xxxx	True	True
1	es	ser	AUX	AUX__Mood=Ind|Number=Sing|Person=3|Tense=Pres|...	cop	xx	True	True
2	una	uno	DET	DET__Definite=Ind|Gender=Fem|Number=Sing|PronT...	det	xxx	True	True
3	frase	frase	NOUN	NOUN__Gender=Fem|Number=Sing	ROOT	xxxx	True	False
4	de	de	ADP	ADP__AdpType=Prep	case	xx	True	True
5	ejemplo	ejemplo	NOUN	NOUN__Gender=Masc|Number=Sing	nmod	xxxx	True	True
6	.	.	PUNCT	PUNCT__PunctType=Peri	punct	.	False	False

spaCy proporciona una tabla resumen de los conceptos que se manejan en NLP, también puede consultarse en la web.

NAME	DESCRIPTION
Tokenization	Segmenting text into words, punctuations marks etc.
Part-of-speech (POS) **Tagging**	Assigning word types to tokens, like verb or noun.
Dependency Parsing	Assigning syntactic dependency labels, describing the relations between individual tokens, like subject or object.
Lemmatization	Assigning the base forms of words. For example, the lemma of "was" is "be", and the lemma of "rats" is "rat".
Sentence Boundary Detection (SBD)	Finding and segmenting individual sentences.
Named Entity Recognition (NER)	Labelling named "real-world" objects, like persons, companies or locations.
Entity Linking (EL)	Disambiguating textual entities to unique identifiers in a Knowledge Base.
Similarity	Comparing words, text spans and documents and how similar they are to each other.
Text Classification	Assigning categories or labels to a whole document, or parts of a document.
Rule-based Matching	Finding sequences of tokens based on their texts and linguistic annotations, similar to regular expressions.
Training	Updating and improving a statistical model's predictions.
Serialization	Saving objects to files or byte strings.

Es importante comprender bien los conceptos que se manejan para poder realizar un correcto análisis del texto. Aunque al principio puede costar, la curva de aprendizaje de spaCy es muy suave y permite gran flexibilidad.

ALGUNAS UTILIDADES ADICIONALES

Rule-based matching:

Los matchers de spaCy permiten encontrar coincidencias de un patrón en el texto, como las expresiones regulares. La gran diferencia es que permite incorporar las predicciones del modelo (como el POS tag o tipo de palabra) al patrón.

```
[19]: from spacy.matcher import Matcher

      matcher = Matcher(nlp.vocab)

      pattern = [{"TEXT": "iPhone"}, {"TEXT": "X"}]
      matcher.add("IPHONE_PATTERN", None, pattern) # el segundo argumento es un callback opcional

      doc = nlp("Acabo de comprar el iPhone X en la tienda de abajo.")

      coincidencias = [doc[start:end] for (match_id, start, end) in matcher(doc)]
      coincidencias

[19]: [iPhone X]
```

```
[21]: pattern = [
          {"LEMMA": "encantar", "POS": "VERB"},
          {"POS": "DET"},
          {"POS": "NOUN"}
      ]

      matcher.add("encantar", None, pattern)

      doc = nlp("Le encantan los libros.")
      coincidencias = [doc[start:end] for (match_id, start, end) in matcher(doc)]
      coincidencias

[21]: [encantan los libros]
```

Constructor de Span y Doc:

Cada modelo nlp y cada Doc tienen un vocabulario aprendido, que se almacena en el objeto Vocab. El Vocab es una tabla de equivalencia de doble sentido entre la cadena de cada palabra y su codificación.

```
[33]: nlp.vocab['café'].orth

[33]: 32833993555699147

[34]: nlp.vocab[32833993555699147].text

[34]: 'café'
```

Cada entrada del vocabulario es un Lexeme, que almacena información de la palabra independiente del contexto en el que se utilice (por ejemplo, la forma de la palabra, si es alfanumérico, etcétera)

Los objetos Doc, Span y Token pueden importarse de spacy.tokens. Para construir un Doc es necesario una lista de palabras con sus espacios y un vocab (del modelo), mientras que para construir un Span es necesario un doc y los índices de los tokens inicial y final, además de poder asignarle una etiqueta opcional.

```
[35]: from spacy.tokens import Doc, Span

      palabras = ["¡", "Hola", "mundo", "!"]
      espacios = [False, True, False, False]

      doc = Doc(nlp.vocab, words = palabras, spaces = espacios)
      doc

[35]: ¡Hola mundo!
```

```
[38]: doc = nlp("Saludos desde el planeta Marte.")

      span = Span(doc, 3, 5)
      span

[38]: planeta Marte
```

Se recomienda trabajar con los Doc y Span y pasar el resultado a cadena lo más tarde posible.

Named entities:

Las entidades son nombres del mundo real, como pueden ser nombres de personas, ciudades, productos, empresas, etcétera. Podemos acceder a las entidades mediante doc.ents, y ver su categoría por label_. Recordar que si no sabemos qué significa cada categoría, podemos utilizar la función explain.

```
[40]: doc = nlp("Mercurio y Venus están más cerca del Sol que la Tierra. Google no opina lo mismo.")
      entidades = [(ent.text, ent.label_) for ent in doc.ents]

      print(*entidades, sep='\n')
```

```
('Mercurio', 'LOC')
('Venus', 'LOC')
('Sol', 'LOC')
('Tierra', 'LOC')
('Google', 'ORG')
```

Las named entities suelen requerir un mayor ajuste para cada caso personalizado, para añadir nuevas entidades relacionadas con empresas que no estén catalogadas, o con productos nuevos. spaCy da mucha libertad y flexibilidad con ese reajuste y configuración.

OPCIONES DE CUSTOMIZACIÓN

Entre las opciones de customización que permite spaCy, podemos destacar:

Modificar el pipeline y agregar nuevas componentes.

El pipeline es la sucesión de funciones que se ejecutan cuando se llama el modelo nlp en un texto. Podemos desactivar (de manera temporal o permanente) algunas de las funciones por defecto y añadir nuevas funciones. La única restricción es que cada función que se añada al pipeline debe aceptar un Doc como argumento y devolver el Doc modificado.

Esto nos permite tener un control total del procesamiento de cada texto, añadiendo nuevas componentes en función de las necesidades de la aplicación. Podemos acceder al pipelien de cada modelo mediante nlp.pipe_names o nlp.pipeline, y añadir las nuevas funciones con nlp.add_pipe(función, before/after/start/end), según el orden donde se quiera añadir la función.

Añadir nuevos atributos, propiedades y métodos a los objetos Token, Span y Doc.

spaCy permite añadir nuevos atributos, propiedades y métodos a cada objeto, que estarán accesibles mediante "._", por ejemplo, token._.esColor. El "_" distingue las componentes del objeto por defecto y las creadas por el usuario. Esto permite añadir cualquier tipo de metadato o información deseable a cada objeto, para un análisis en mayor profundidad.

Ajuste de modelos / creación de un modelo desde 0.

Con spacy.blank("es") generamos un modelo nuevo de idioma español. El modelo viene sin pipeline así que debe añadirse, sea el pipeline por defecto o una personalizada. Con nlp.begin_training() inicializamos los pesos de la red neuronal para modelos nuevos, mientras que con nlp.update() entrenamos o actualizamos el modelo con los datos de entrenamiento proporcionados. Cabe destacar que el reentrenamiento o ajuste de un modelo

preentrenado puede resultar en el "olvido" de características aprendidas previamente si no se le proporcionan datos de entrenamiento adecuados.

CASOS DE USO

spaCy se puede utilizar para crear aplicaciones que requieran procesamiento de lenguaje natural. Esto incluye, pero no se limita a:

- Resúmenes automáticos de textos.
- Reconocimiento de entidades nombradas.
- Sistemas de preguntas y respuestas.
- Análisis de sentimientos.

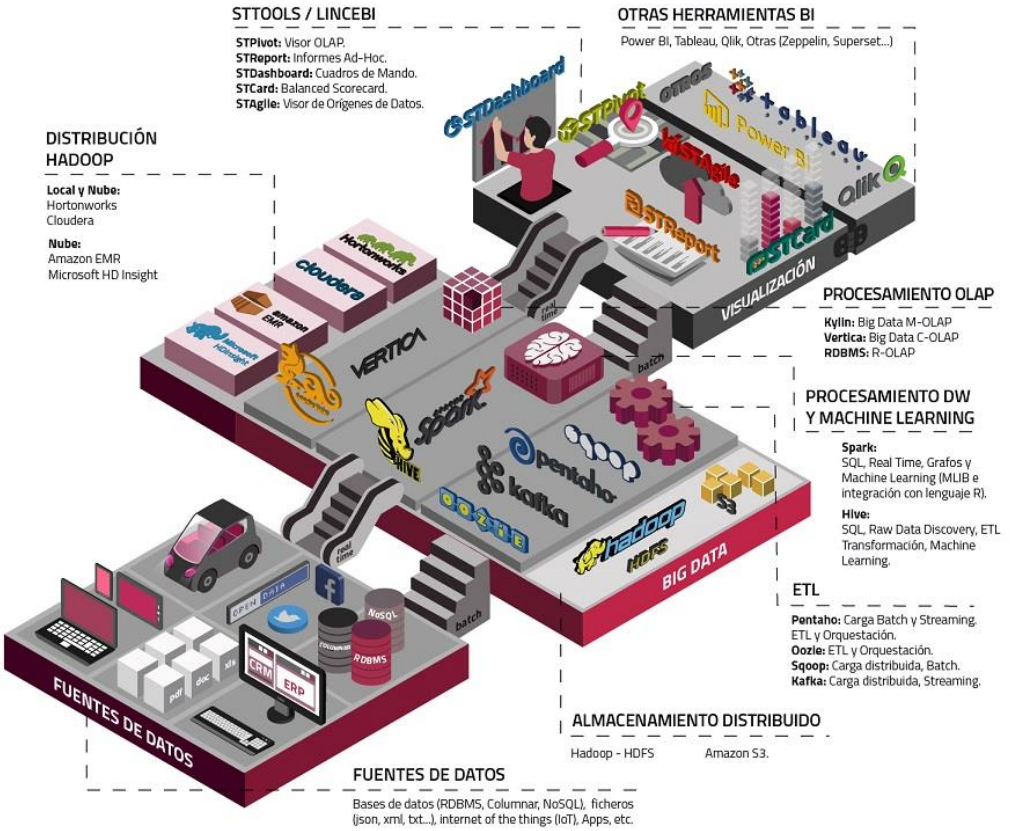
CONCLUSIÓN

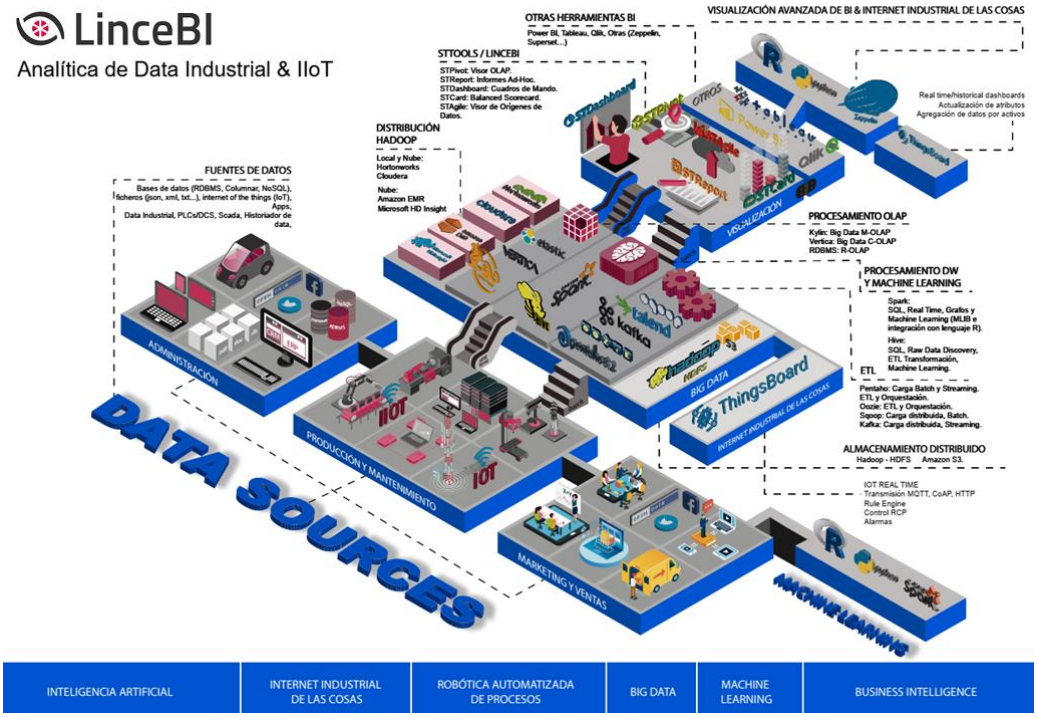
Entender bien el funcionamiento interno es imprescindible para poder crear una aplicación a medida, pero la documentación proporciona transparencia en este aspecto, y gracias a la sintaxis y a funciones de utilidad como explain o displacy, la curva de aprendizaje es muy suave, a diferencia de otras herramientas de procesamiento de lenguaje natural, como puede ser Tensorflow.

spaCy es una librería orientada al desarrollo y producción de aplicaciones que requieran de procesamiento de lenguaje natural. Las posibilidades de configuración de los modelos, además de proporcionar modelos preentrenados hacen de spaCy una gran opción cuando se busca un framework con este fin, dado que la libertad de configuración hace posible la creación de aplicaciones con casi cualquier objetivo de análisis de texto.

TECNOLOGÍAS

Trabajamos con las principales tecnologías y somos Partners Certificados de Vertica, Talend, Microsoft, Snowflake, Kylligence, Pentaho, etc.


STTOOLS / LINCEBI
STPivot: Visor OLAP.
STReport: Informes Ad-Hoc.
STDashboard: Cuadros de Mando.
STCard: Balanced Scorecard.
STAgile: Visor de Orígenes de Datos.
OTRAS HERRAMIENTAS BI
Power BI, Tableau, Qlik, Otras (Zeppelin, Superset...)
DISTRIBUCIÓN HADOOP
Local y Nube:
Hortonworks
Cloudera
Nube:
Amazon EMR
Microsoft HD Insight
VISUALIZACIÓN
PROCESAMIENTO OLAP
Kylin: Big Data M-OLAP
Vertica: Big Data C-OLAP
RDBMS: R-OLAP
PROCESAMIENTO DW
Y MACHINE LEARNING
Spark:
SQL, Real Time, Grafos y Machine Learning (MLIB e integración con lenguaje R).
Hive:
SQL, Raw Data Discovery, ETL Transformación, Machine Learning.
BIG DATA
ETL
Pentaho: Carga Batch y Streaming. ETL y Orquestación.
Oozie: ETL y Orquestación.
Sqoop: Carga distribuida, Batch.
Kafka: Carga distribuida, Streaming.
FUENTES DE DATOS
ALMACENAMIENTO DISTRIBUIDO
Hadoop - HDFS
Amazon S3.
FUENTES DE DATOS
Bases de datos (RDBMS, Columnar, NoSQL), ficheros (json, xml, txt...), internet of the things (IoT), Apps, etc.


LinceBI
Analítica de Data Industrial & IIoT
OTRAS HERRAMIENTAS BI
VISUALIZACIÓN AVANZADA DE BI & INTERNET INDUSTRIAL DE LAS COSAS
STTOOLS / LINCEBI
DISTRIBUCIÓN HADOOP
FUENTES DE DATOS
PROCESAMIENTO OLAP
PROCESAMIENTO DW Y MACHINE LEARNING
ETL
ALMACENAMIENTO DISTRIBUIDO
DATA SOURCES
IIOT
IOT
ThingsBoard
INTELIGENCIA ARTIFICIAL
INTERNET INDUSTRIAL DE LAS COSAS
ROBÓTICA AUTOMATIZADA DE PROCESOS
BIG DATA
MACHINE LEARNING
BUSINESS INTELLIGENCE

INFORMACIÓN SOBRE STRATEBI

Stratebi es una empresa española, con sede en Madrid y oficinas en Barcelona, Alicante y Sevilla, creada por un grupo de profesionales con amplia experiencia en sistemas de información, soluciones tecnológicas y procesos relacionados con soluciones de Open Source y de inteligencia de Negocio.

Esta experiencia, adquirida durante la participación en proyectos estratégicos en compañías de reconocido prestigio a nivel internacional, se ha puesto a disposición de nuestros clientes.

Somos **Partners Certificados en Microsoft PowerBI** con una dilatada experiencia

Stratebi es la única empresa española que ha estado presente todos los Pentaho Developers celebrados en Europa habiendo organizado el de España.

En Stratebi nos planteamos como **objetivo** dotar a las compañías e instituciones, de herramientas escalables y adaptadas a sus necesidades, que conformen una estrategia Business Intelligence capaz de rentabilizar la información disponible. Para ello, nos basamos en el desarrollo de soluciones de Inteligencia de Negocio, mediante tecnología Open Source.

Stratebi **son** profesores y responsables de proyectos **del Master en Business Intelligence de la Universidad UOC, UCAM, EOI…**

Los profesionales de Stratebi son los creadores y autores del primer weblog en español sobre el mundo del Business Intelligence, Data Warehouse, CRM, Dashboards, Scorecard y Open Source. Todobi.com

Stratebi es partner de las principales soluciones Analytics: Microsoft Power BI, Talend, Pentaho, Vertica, Snowflake, Kyligence, Cloudera…

Todo Bi, se ha convertido en una referencia para el conocimiento y divulgación del Business Intelligence en español.

OTROS

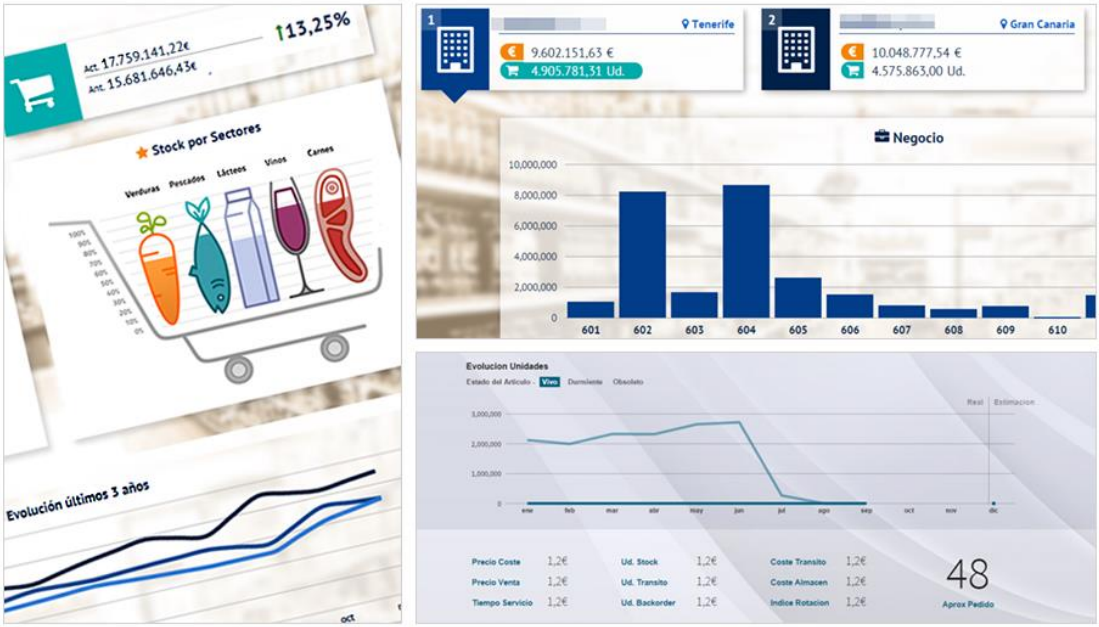
Trabajamos en los principales sectores y con algunas de las compañías y organizaciones más importantes de España.

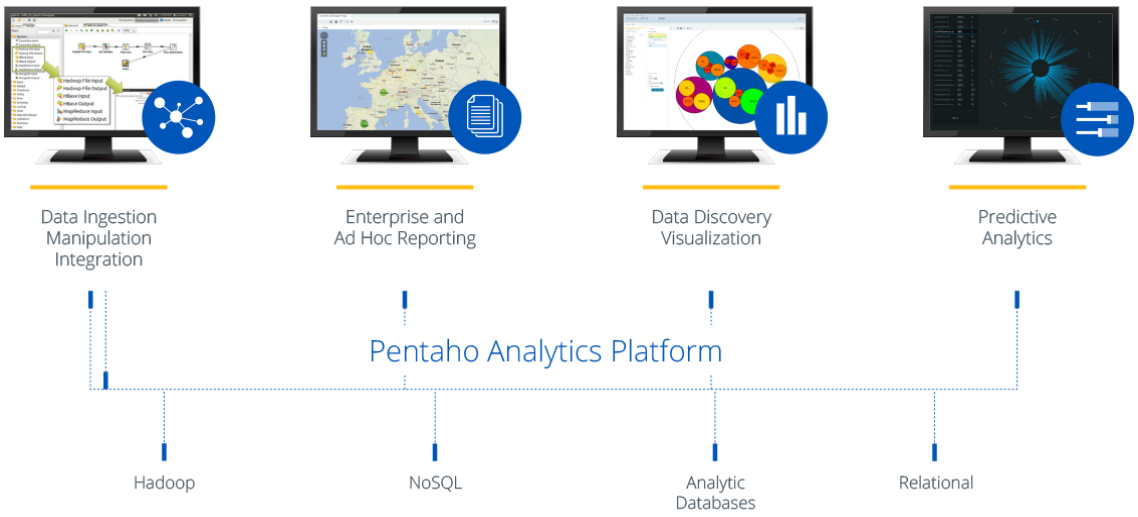
SECTOR PRIVADO

SECTOR PÚBLICO

EJEMPLOS DE DESARROLLOS ANALYTICS

A continuación, se presentan **ejemplos de algunos screenshots** de cuadros de mando diseñados por Stratebi, con el fin de dar a conocer lo que se puede llegar a obtener, así como Demos Online en la web de Stratebi:

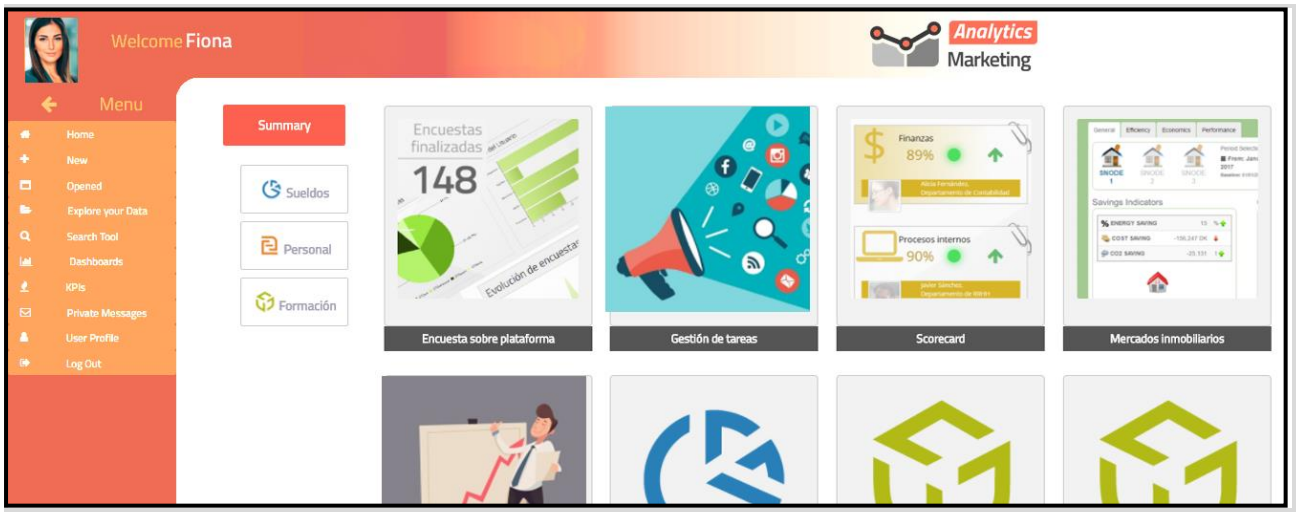





facebook DASHBOARD COMENTARIOS
Sexo
65% 35%
Sentimiento
28%
60%
12%
Localización
Tipo de recurso
Fotos
Videos
Status
Preguntas
Evolución Comentarios por días
facebook DASHBOARD COMENTARIOS
Sexo
65% 35%
Sentimiento
28%
60%
12%
Localización
Tipo de recurso
Evolución Comentarios por días


smart MADRID
Última Actualización
18:24
TOP 5 PUNTOS DE MEDIDA
PROXIMIDAD DE PUNTOS DE MEDIDA
EVOLUCIÓN INTENSIDAD DE TRÁFICO


Procedencia
FUENTE-ÁLAMO
Sexo
50%
Masculino
50%
Femenino
Rango de Edad
18-20
20-22
22-24
24-26
26-28
30-35
28-30
35-40
GRADO EN BIOLOGÍA
Créditos Matriculados: 6.500
Buscar Plan de Estudio
INTERÉS PLATAFORMA
181 Inscritos en Cursos
8 Desinscritos en Cursos
181 Inicios de Cursos
110 Inscritos en Eventos Virtuales
63 Inscritos en Debates
181 Inscritos en Talleres
USUARIOS ACTIVOS POR PAÍS
INSCRITOS EN CURSOS POR COMUNIDAD EDU
JORNADA COMPLETA
IDIOMAS
TIC
TIPO DE CENTRO
TIPO DE AUSENCIA
Nº de profesores por sexo
36%
64%
Nº total de profesores
1.587
Media de edad
43 años
Búsqueda


Welcome Fiona
Analytics
Marketing
Menu
Home
New
Opened
Explore your Data
Search Tool
Dashboards
KPIs
Private Messages
User Profile
Log Out
Summary
Sueldos
Personal
Formación
Encuestas finalizadas
148
Evolución de encuestas
Encuesta sobre plataforma
Gestión de tareas
Finanzas
89%
Procesos internos
90%
Scorecard
Savings Indicators
Mercados inmobiliarios