

2018

# Pràctica 3 - Spark

BASES DE DADES AVANÇADES  
PAU SANCHEZ I GUILLEM RABIONET

UNIVERSITAT DE BARCELONA | Semestre de tardor

## 1. Neteja del text del Quijote i reconstrucció dels paràgrafs

El primer que cal fer és netejar el text del Quijote i eliminar els caràcters estranys i símbols del text, així com substituir aquelles vocals que accentuades per elles mateixes sense accentuar.

Per dur a terme aquesta tasca s'ha creat un programa en llenguatge Scala. Aquest rep el fitxer del Quijote sense cap mena de tractament i retorna un fitxer amb les següents característiques:

- Els accents han estat eliminats (substituïts per la vocal sense accent)
- Cada línia de text és un paràgraf sencer (s'han reconstruït els paràgrafs)
- S'han filtrat les paraules buides (aquelles paraules que no aporten valor)

Nota: Tot i que l'enunciat no ho demanava, s'ha cregut que no tenia sentit compara paràgrafs donant valor a articles, pronoms febles, determinants, etc. Aquestes s'anomenen stop words i han estat eliminades del text per a poder fer un anàlisi i comparació de paraules més rigorós.

Aquest programa de neteja és l'encarregat de generar l'Output que serà absorbit per DataBricks i amb ell es crearà la taula de dades per a fer l'entrenament del model i la corresponent vectorització dels paràgrafs.

## 2. Com executar el programa de neteja

El programa està fet en Scala i per executar-lo cal tenir Scala instal·lat, si el lector no té instal·lat Scala pot fer-ho molt fàcilment amb la següent instrucció:

```
>> sudo apt-get update
```

```
>> sudo apt-get install scala
```

Una vegada instal·lat scala només cal executar l'script.sh que s'ha preparat. Ell mateix fa la compilació, l'execució i després elimina els fitxers '.class' de compilació d'Scala. L'script es troba a la següent ruta relativa al projecte:

>> /word\_processing/script.sh (Executar des de dins la carpeta, són rutes relatives!)

Vegem doncs aquest script al detall

```
1 echo "Compilant el fitxer i creant executables.."
2 scalac CleanText.scala
3 echo "Netejant els textos.."
4 scala CleanText
5 echo "Eliminant els fitxers de compilació '.class'"
6 rm *.class
7 echo "Ara el fitxer quiijote net és a la ruta: '../files/quijote/clean_quijote.txt'"
```

Com es pot veure el programa que s'ha executat es diu CleanText.scala. Aquest és el resultat:

(Executar des de dins la carpeta, són rutes relatives!)

```
pausanchezv@pausanchezv: ~/Spark_Word2Vec_TensorFlow/word_processing
pausanchezv@pausanchezv:~/Spark_Word2Vec_TensorFlow/word_processing$ ./script.sh
Compilant el fitxer i creant executables..
Netejant els textos..
Eliminant els fitxers de compilació '.class'
Ara el fitxer quiijote net és a la ruta: '../files/quijote/clean_quijote.txt'
pausanchezv@pausanchezv:~/Spark_Word2Vec_TensorFlow/word_processing$ |
```

El programa sencer es pot veure fàcilment accedint al fitxer '/word\_processing/CleanText.scala', amb qualsevol editor com per exemple 'kate' o 'sublime text'. Consta d'una classe de tipus Object d'Scala, que intenta emular un Singleton de qualsevol altre llenguatge orientat a objectes. El seu mètode principal es diu 'main' i s'executa de manera automàtica. A partir d'aquesta crida es desencadena la neteja dels textos i posterior generació del fitxer. Ho fa de la següent manera:

- Llegeix el fitxer: './files/quijote/raw\_quijote.txt'
- Fa el processat
- Genera el nou fitxer: './files/quijote/clean\_quijote.txt'

Per executar les tasques anteriors la classe disposa e 4 funcions bàsiques:

- filterStopWords:: per excloure les paraules buides
- cleanStrings:: per netejar les línies de caràcters estranys i accents
- constructParagraphs:: genera els paràgrafs a partir de les línies de text
- main:: distribueix les tasques repartint-les a les funcions anteriors i genera l'output

Vegem l'aspecte que prenen els textos una vegada processats i havent aplicat el programa de neteja (cada número de línia representa un paràgraf):

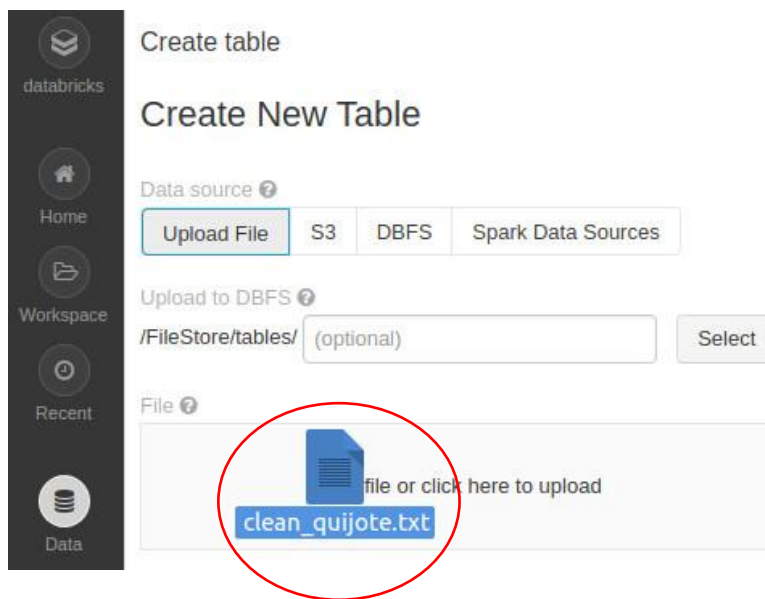
```
1 ingenioso hidalgo don quijote mancha
2 tasa
3 yo juan gallo andrada escribano camara rey senor residen consejo certifico doy fe que habiendo visto senores del libro intitulado ingenioso
hidalgo mancha compuesto miguel cervantes saavedra tasaron pliego dicho libro tres maravedis medio ochenta tres pliegos dicho precio monta dicho
libro docientos noventa maravedis medio vender papel dieron licencia precio pueda vender mandaron tasa ponga principio dicho libro pueda vender
ella y dello conste di presente valladolid veinte dias mes diciembre mil seiscientos quatro anos
4 juan gallo andrada
5 testimonio erratas
6 libro cosa digna corresponda original testimonio haber correcto di fee colegio madre dios teologos universidad alcala diciembre 1604 anos
7 licenciado francisco murcia llana
8 rey
9 quanto parte vos miguel cervantes fecha relacion habiades compuesto libro intitulado ingenioso hidalgo mancha habia costado mucho era util
provechoso pedistes suplicastes mandasemos dar licencia facultad le imprimir privilegio fuesemos servidos o merced fuese visto consejo quanto
dicho libro hicieron diligencias premativa ultimamente fecha impresion libros dispone acordado debiamos mandar dar cedula vos dicha razon
```

A la pràctica es dóna el fitxer generat, però per comprovar que tot funciona perfectament, el lector pot esborrat el fitxer './files/quijote/clean\_quijote.txt', tornar a executar l'script.sh i comprovar que el fitxer s'ha generat una altra vegada.

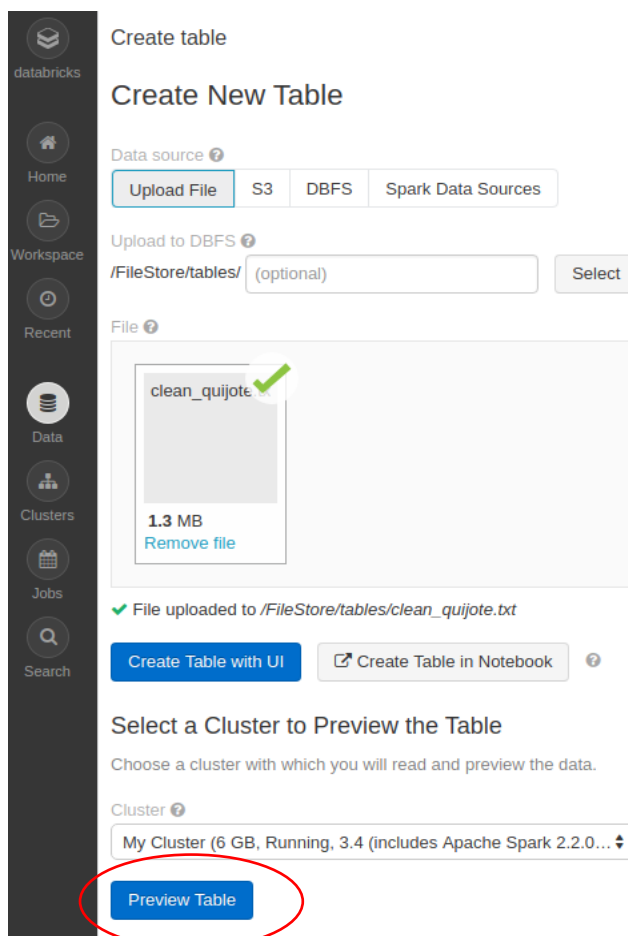
### 3. Importació del Quijote a DataBricks

Una vegada obtingut el Quijote processat, net de símbols i construït per paràgrafs cal importar-lo a DataBricks per construir la taula que permetrà fer la lectura del fitxer des del notebook.

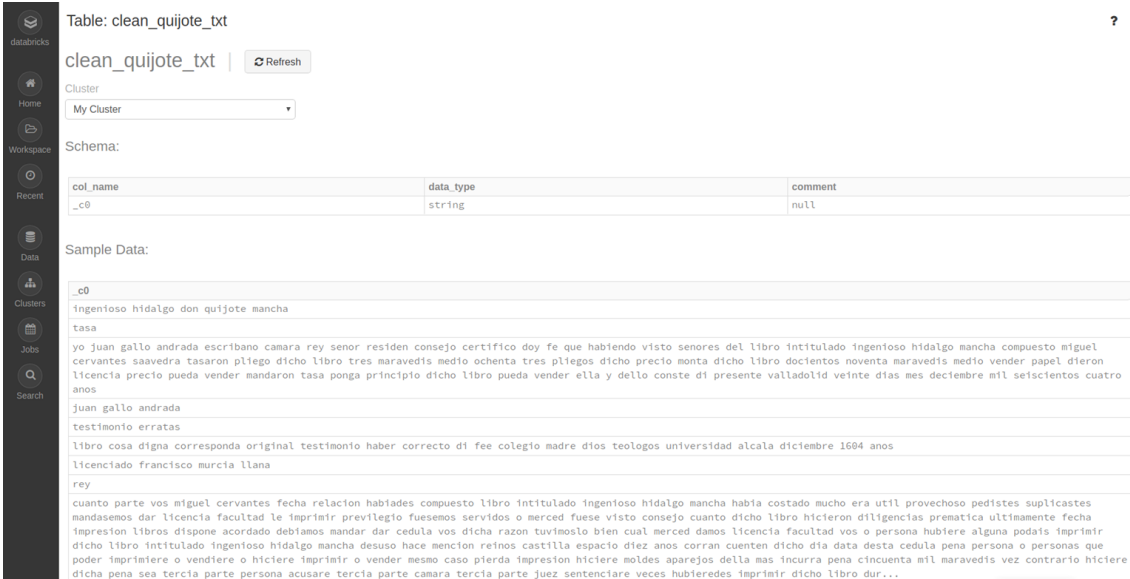
El primer que cal fer és dirigir-se a l'apartat de creació d'una nova taula i arrossegar el fitxer 'clean\_quijote.txt' obtingut al pas anterior:



Una vegada s'ha carregat el fitxer cal prémer el botó de creació de taula mitjançant UI i acte seguit seleccionar el cluster de 6gb que DataBricks ofereix a l'usuari.



Una vegada s'ha creat la taula, aquesta pren el següent aspecte:



The screenshot shows the Databricks web interface. On the left is a sidebar with navigation icons for Home, Workspace, Recent, Data, Clusters, Jobs, and Search. The main area displays the table 'clean\_quijote\_txt' with a 'Refresh' button. Below the table name, it shows the cluster 'My Cluster'. The schema is defined as follows:

| col_name | data_type | comment |
|----------|-----------|---------|
| _c0      | string    | null    |

Below the schema, the 'Sample Data:' section shows a preview of the table's content, including fields like '\_c0', 'ingenioso hidalgo don quijote mancha', 'tasa', and various text entries from the book 'Don Quixote'.

Aquest és el pas intermig que cal fer per importar un fitxer del disc local i poder-lo utilitzar a DataBricks, en un format que aquest pugui entendre.

Ara el nostre fitxer 'clean\_quijote.txt' es troba també a la ruta relativa al projecte de DataBricks: '/FileStore/tables/clean\_quijote.txt', a més d'estar a la ruta del nostre projecte local a disc. Per tant per llegir el fitxer des del notebook l'única cosa que cal fer és invocar la següent crida:

```
sc.textFile("/FileStore/tables/clean_quijote.txt")
```

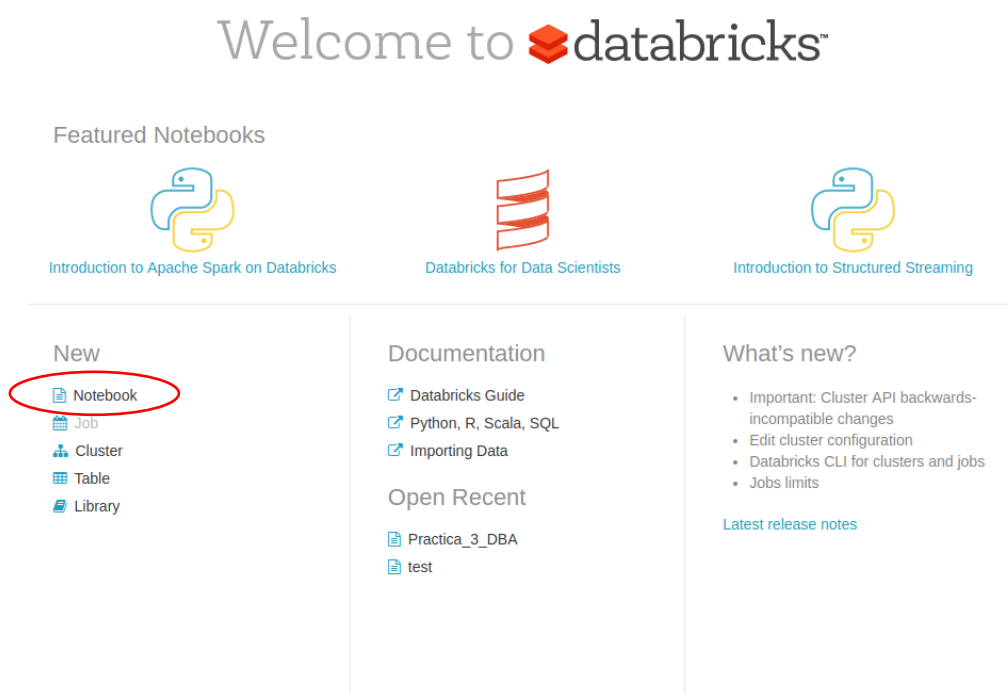
## 4. Llenguatge de desenvolupament

Una vegada més, el llenguatge triat pel desenvolupament de la part de la pràctica que s'explica ha continuació, ha estat Scala, ja que aquest és el llenguatge és el llenguatge mestre per combinar amb Spark. Val a dir que Scala és a Spark allò que Java és a Android. Altrament també s'hagués pogut desenvolupar amb Python, que disposa del Framewrok PySpark per tractar Spark de manera igualment òptima, o també Java.

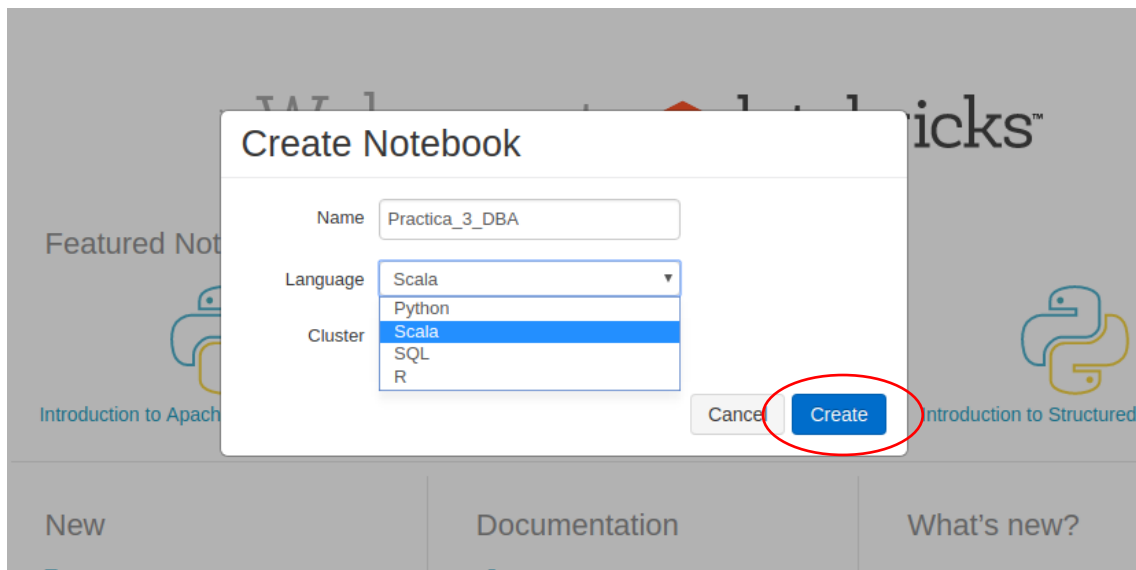
Scala es caracteritza per ser un llenguatge compilat, funcional i orientat a objectes (python té les mateixes característiques excepte que és interpretat en lloc de compilat), i a la pràctica s'ha utilitzat tant com s'ha pogut el poder dels llenguatges funcionals (maps, filters, etc..) així com les característiques dels llenguatges orientats a objectes.

## 5. Creació i execució del notebook a DataBricks

Per crear el notebook de treball a DataBricks cal dirigir-se a la pàgina de DataBricks Community i logar-se al sistema, una vegada fet això, trobem les següents opcions de treball:



Com no podia ser d'altra manera, cal triar l'opció 'New → Notebook' de la part esquerra de la pantalla. Aquest acte ens obre el següent ventall de possibilitats:



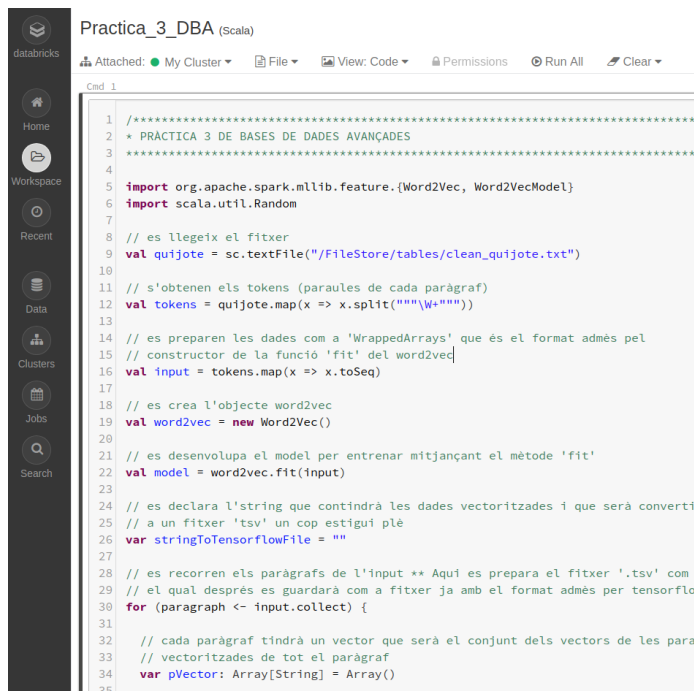
Després de posar nom a la pràctica, cal triar el llenguatge Scala i clicar el botó 'create'. Això portarà al lector directament al nou notebook anomenat 'Practica\_3\_DBA'. Però aquest notebook està buit!

El que cal fer ara és dirigir-se a la ruta relativa al projecte `'/files/notebook_databricks/notebook.txt'` i copiar tot el contingut del fitxer per enganxar-lo tot a la primera cel·la del notebook. (El notebook només farà servir la primera cel·la, doncs n'hi ha prou!). Per tant els passos a seguir són:

- Accedir a la ruta relativa `'/files/notebook_databricks/notebook.txt'` i obrir el fitxer `'.txt'`, amb un editor qualsevol, per exemple, sublime, kate, etc..
- Copiar tot el contingut del fitxer: `'ctrl + a'` per seleccionar tot el text i `'ctrl + c'` per a copiar-lo.
- Dirgir-se ala primera cel·la del notebook que s'ha creat al pas anterior
- Enganxar (`ctrl + v`) el contingut del fitxer dins la cel·la
- Executar la cel·la `'ctrl + intro'` o botó 'play' de la interfície gràfica

El resultat dels passos anteriors hauria de ser com el següent, i s'hauria de visualitzar el notebook d'aquesta manera:





```

1 /*****
2 * PRACTICA 3 DE BASES DE DADES AVANÇADES
3 *****/
4
5 import org.apache.spark.mllib.feature.{Word2Vec, Word2VecModel}
6 import scala.util.Random
7
8 // es llegeix el fitxer
9 val quijote = sc.textFile("/FileStore/tables/clean_quijote.txt")
10
11 // s'obtenen els tokens (paraules de cada paràgraf)
12 val tokens = quijote.map(x => x.split("\\W+"))
13
14 // es preparen les dades com a 'WrappedArrays' que és el format admès pel
15 // constructor de la funció 'fit' del word2vec
16 val input = tokens.map(x => x.toSeq)
17
18 // es crea l'objecte word2vec
19 val word2vec = new Word2Vec()
20
21 // es desenvolupa el model per entrenar mitjançant el mètode 'fit'
22 val model = word2vec.fit(input)
23
24 // es declara l'string que contindrà les dades vectoritzades i que serà convertit
25 // a un fitxer 'tsv' un cop estigui ple
26 var stringToTensorFlowFile = ""
27
28 // es recorren els paràgrafs de l'input ** Aquí es prepara el fitxer '.tsv' com
29 // el qual després es guardarà com a fitxer ja amb el format admès per tensorflow
30 for (paragraph <- input.collect) {
31
32   // cada paràgraf tindrà un vector que serà el conjunt dels vectors de les para
33   // vectoritzades de tot el paràgraf
34   var pVector: Array[String] = Array()
35

```

## 6. Detall del programa (Part 1: Com fer l'execució)

Abans d'entrar en el detall del programa cal explicar un aspecte important al lector:

L'objectiu del programa és generar un output en format 'tsv' que és l'admes per TensorFlow a la darrera part de la pràctica. Com que no s'ha trobat la manera d'exportar un fitxer a disc local des de la web de DataBricks ha calgut 'buscar-se la vida', i s'ha fet de la següent manera: el programa en DataBricks genera un string de sortida. Aquest string representa el contingut del que hauria de ser el fitxer de sortida 'tsv'. D'aquesta manera el lector pot veure el correcte funcionament del programa, i per no fer-ho molt farragós, es treuen per pantalla només els resultats dels 10 primers paràgrafs: paràgraf + vector de característiques del paràgraf a cada línia.

Ara bé, com que cal tenir el fitxer 'tsv' per representar tots els paràgrafs, s'ha optat per la següent solució:

- S'ha creat un projecte de IntelliJ Idea que s'adjunta a la pràctica amb exactament el mateix codi que conté el notebook, tret de l'última línia del codi, que enlloc de fer un simple `'println'`, el que fa és gravar el fitxer a disc, i ho fa a la següent ruta relativa al projecte: `'/files/quijote/quijote_to_tensorflow.tsv'`

Un aspecte important és que aquest fitxer `'tsv'` ja s'entrega generat a la pràctica, ara bé, si el lector vol comprovar el bon funcionament del programa només cal que executi el projecte d'IntelliJ, el qual també s'ha adjuntat degudament. De la mateixa manera el lector pot executar el notebook per veure les 10 primeres sortides impreses en el format que espera TensorFlow.

Un altre detall important és que per poder visualitzar correctament les dades a TensorFlow (més endavant se'n parla amb detall), s'ha preferit generar el fitxer d'output pels 1000 primers paràgrafs del text (el quijote en té als voltants de 5000), doncs la visualització de tots els paràgrafs a TensorFlow quedava 'massa plena', es barrejava i se solapava la informació i no s'entenia res de res a la visualització. Per aquest motiu s'ha decidit 'capar' l'output `'tsv'` a 1000 línies on cada línia conté el text del paràgraf i un vector de característiques que el representa.

Nota: Si el lector té el desig comprovar la visualització amb tots els paràgrafs enlloc dels primers 1000, només ha de canviar la instrucció `'input.take(1000)'` per `'input.collect'` i re-executar per generar el nou output que es trobarà a la mateixa ruta havent sobreescrit l'anterior. Ara bé, s'avisa que la visualització de 5000 paràgrafs és absolutament nefasta. Molt millor amb 1000.

Una vegada entès això, es pot passar al desenvolupament de l'entrenament i la vectorització del text, al següent apartat.

## 7. Detall del programa (Part 2: Desenvolupament)

El programa, que s'explica a continuació es pot trobar a una de les següents rutes relatives:

```
>> /files/notebook_databricks/notebook.txt
```

```
>> /Intellij_Word2vec/Word2vec_P3_DBA/src/main/scala/Main.scala
```

S'aconsella tenir obert un dels dos fitxers anteriors (preferiblement la segona ruta que és la que desa el fitxer d'output a disc) per seguir l'explicació i el detall del programa, que s'explica a continuació:

- Es llegeix el fitxer del quijote net de símbols i ordenat per paràgrafs mitjançant el context d'Spark
- De cada paràgraf s'obtenen els tokens, és a dir, ara cada paràgraf és un vector de paraules (recordar que s'han filtrat les paraules buides al programa de neteja).
- Es transformen els vectors en 'WrappedArrays' que és el format de vector admès per la funció 'fit' de Word2Vec.
- Es crea l'objecte word2vec i es crida al mètode 'fit' que rep com a argument l'input generat de 'WrappedArrays'. Aquesta funció desenvolupa i entrena el nostre model, que més endavant es farà servir per dur a terme la vectorització de les paraules.

Aquí cal fer un incís i explicar com es farà la vectorització abans de seguir amb l'execució del programa:

*Quan es vectoritza una paraula s'obté la paraula seguida d'un vector de característiques que la representen. Aquestes característiques són valors numèrics de tipus flotant, i per defecte se'n generen 100 per cada paraula. D'això se n'encarrega el mètode transform que es crida amb l'objecte del model. Ara cada paraula vectoritzada té una forma com la següent:*

*A [float, float, float, float ..... float] → On A és la paraula i cada float és un valor numèric en coma flotant que representa una de les 100 característiques de la paraula.*

*El que cal, però, aconseguir, és un vector representatiu d'un paràgraf ja que el que es pretén comparar són paràgrafs sencers i no paraules! Aquest acabarà tenint la següent forma:*

*ABCDEFGH [float, float, float, float ..... float] → On ABCDEFGH són les paraules del paràgraf i cada float és una característica, atenció, que representa una de les 100 característiques d'entre les característiques de les paraules de tot el paràgraf.*

*Per generar el vector de cada paràgraf el primer que s'ha fet és generar un vector gegant per cada paràgraf, amb les característiques de totes les paraules que el componen. Per exemple un paràgraf de 30 paraules, de moment, tindrà 3000 característiques. Ara és quan arriba el problema: TensorFlow no admet vectors de característiques de diferent mida, sinó que només reconeix fitxers de vectors de característiques simètrics (vectors de mida idèntica). Com que les característiques de cada paraula són 100, això fa que que s'hagin de 'capar' les característiques del paràgraf a 100, per tractar el cas en que un paràgraf només està compost per una paraula (es donen uns quants casos).*

*Ara cal triar quina és l'estratègia per, un cop vectoritzades les paraules, crear el vector de característiques representatiu del paràgraf. El que s'ha fet ha estat seleccionar aleatòriament 100 característiques del gran vector de característiques del paràgraf i així s'obté el vector reduït de característiques que representen totes les paraules del paràgraf. Aquesta és una manera de fer-ho però de ben segur que n'hi ha d'altres.*

*Una vegada s'ha entès això, es pot seguir amb l'execució de l'algorisme allà on s'havia deixat:*

- Es crea un string buit on s'aniran afegit les línies: text del paràgraf + vector de característiques correctament tabulat i alineat. Després només caldrà desar aquest string en un fitxer d'extensió .tsv per ser executat a TensorFlow.
- Es recorren les paraules de cada paràgraf
- Es vectoritza cada paraula 'model.transform(word)'
- Es va construint el gran vector de característiques del paràgraf
- Se seleccionen 100 característiques de manera aleatòria
- a l'últim pas poden passar dues coses depenent d'on tingui lloc l'execució:
- Si s'executa al notebook es mostrarà per pantalla el resultat
- Si s'executa al IntelliJIdea es generarà el fitxer '.tsv' (l'entrega ja el dona generat per facilitar la feina al lector).

Visualització del fitxer .tsv que s'ha generat:

```
1 'ingenioso hidalgo don quijote mancha' -0.06559880763593674 -0.03993431478738785 -0.12713152170181274 0.10059939324855804
0.1846388578414917 -0.0198502317070961 0.08566941499710083 0.031083374606370926 -0.012406990863382816 -0.01489638164639473
0.015904521566069642 -0.23454971611499706 0.12329215556383133 0.016835425049066544 0.11433728784322739 -0.009318018332123756
-0.08119073510169983 0.008635982871055603 0.19456037878990173 -0.29167935252189636 0.0014076397055760026 -0.0967077910900116
0.034474603831768036 0.22806861865520477 0.015380872413516045 -0.028059672564268112 0.12376580387353897 -0.0025908176321536303
0.057822369039058685 0.16570353507995605 0.016673846170306206 -0.03163042291998863 0.09691055119037628 0.027095599693026543
0.06440228223800659 -0.003038905095309019 -0.030246281996369362 0.017391685396432877 -0.10683951526880264 -0.02309814654290676
-0.0074501470662653345 -0.34403637051582336 0.031234819442033768 -0.009594878181815147 -0.19191357493400574 -0.39992913603782654
-0.0379662849009037 0.0035323956981301308 -0.08754094690084457 0.006693823728710413 0.09636905044317245 0.1259372979402542
-0.03298759568294144 -0.10594423115253448 -0.30060580372810364 0.02057310752570629 -0.022030439227819443 -0.1315014511346817
0.10543674975633621 -0.06229244917631149 -0.2185678482055664 0.04983796551823616 0.03718335540037155 0.15067830681800842
-0.30992293378784912 -0.116594597697258 -0.015395759604871273 0.00966790784150362 -0.12380920350551605 0.004143598489463329
-0.15967422723770142 0.00567600829526782 -0.022890223190188408 -0.14607739448547363 -0.3472483158111572 0.05266387388110161
0.027234364300966263 0.13647514581680298 0.004895161371678114 -0.12152967602014542 0.03139462321996689 0.1569075882434845
-0.017475849017500877 -0.20763391256332397 -0.12293805181980133 -0.350437194108963 0.010588649660348892 0.02669462189078331
0.0664299726486206 0.42234522104263306 0.32698220014572144 -0.15173019468784332 -0.004354275297373533 0.12597692012786865
-0.31023067235946655 -0.023127812892198563 -0.035779885947704315 0.015267944894731045 -0.0335439071059227 -0.14816097915172777
-0.007595054339617491 0.001705997041426599 0.015011598356068134 0.012663651738643646 0.017369238659739494
0.001834783754844546 0.02337516099214554 0.014162867230014766 0.558408495038748E-4 0.018181810155510902 -0.015976205468177795
-0.009019205584932267 0.02764459367471695 0.0020593069112432003 0.0036209714827388525 0.02575280518624306 -0.048384377866785736
0.014909674091339111 0.024428001611065865 -0.013686326332380308 0.023212237283587456 0.004324577824920416 0.017038416451215744
-0.015701016411185265 0.010323682799935341 0.001553498871140182 0.0012988016132599115 -0.0020064765121787786 0.01593875139951706
0.03431906923651695 0.0021264434326440096 0.02118963934481144 0.04125577211380005 0.03323978930711746 0.0413951837792969
-0.0021594827994704247 -0.03923272341489792 0.037903185933828354 -0.017469527199864388 -0.004428728949278593 0.021359562873840332
0.007302590645849705 0.009159897454082966 -0.01909971982240677 -0.0023152625653892756 -0.008582131005823612 -0.03274713456630707
0.005530982743948698 0.00909006129950285 -0.012259173206984997 2.71801371127367E-4 0.013371245935559273 0.01303182635453236
-0.01775076426565647 -0.03275378793478012 0.010559302754700184 0.009967944584786892 0.0013585735578089952 -0.0101626501506567
0.030478836968541145 0.019385134801268578 0.005556157324463129 0.007702736649662256 0.008444219827651978 0.017702186480164528
0.004667859990149736 0.010493047535419464 -0.003789227921515703 0.009838896803557873 -0.01205467339605093 0.027687089517712593
0.012994782067835331 -0.03260188177227974 -0.007162762805819511 0.017000075429677963 -0.02810611017048359 0.003843789454549551
0.04313478618860245 0.0018517673015594482 0.016295043751597404 7.30699161067605E-4 0.0024529576767235994 0.02074996754527092
-0.006826791912317276 2.1492299856618047E-4 -0.004360088612884283 -0.03164013475179672 -0.03388252109289169 -0.019729776307940483
-0.01446246076375246 0.018633795902132988 -0.010931205004453659 -0.01056857779622078 -0.02497611753642559 -0.005130690522491932
0.0038477862253785133 -0.007389997597783804 -0.006824190262705088 0.022473447024822235 0.011307883076369762
3 'yo juan gallo andrada escribano camara rey senor residen consejo certificado doy fe que habiendo visto senores del libro intitulado ingenioso
hidalgo mancha compuesto Miguel cervantes saavedra tataron dicho libro tres maravedis medio ochenta tres pliegos dicho precio pueda dicho
libro dicho mancha compuesto maravedis medio vender papel dieron licencia precio pueda vender mandaron tasa ponga principio dicho libro pueda vender
elta y dello conste di presente valladolid veinte dias mes diciembre mil seiscientos quatro anos' 0.04066690802574158 -0.04767692796886E-4
-0.04144100844860077 0.04446012154221535 0.020500006154179573 0.0014175968244671822 -0.0020920840853886127 -0.010152011767263548
0.048543769866223104 0.044863410256803636 -0.010196038521826267 0.011070232838302258 0.005834800656884909 0.02632856834679842
-0.15982066094875336 -0.034715574232474045 -0.0081421778878188133 -0.025281406939029694 0.03800467702104019 -0.006534866522997618
0.07318318006141663 -0.0441557951271534 -0.0592722938001413345 0.09787116018915176 -0.06423138827085495 -0.0741827389339447
-0.002266984200105071 0.04662945494055748 -0.009841944091022815 0.01452734787017107 0.04398822784423828 0.11917462199926376
0.007177483290433884 -0.013276400044560432 0.01923993229866028 0.04099006578326225 0.04826314374804497 -0.006181388162076473
-0.04826314374804497 -0.003896040376275778 0.019808102399110794 0.0037170341572761536 0.01575104147195816 0.02135753445327282
0.024088139967918396 -0.0647464320063591 -0.0318465493619442 0.05155076086521149 0.007724205031991005 -0.0679703950881958
0.015670148422941566 0.01770969107747078 -0.03834860026836395 0.0057150861248373985 0.004538459703326225 0.027644578367471695
0.04739711433649063 -0.004827884025871754 0.008237620815634727 -0.042471375316381454 -0.010921099223196507 -0.008752713911235332
-0.03559844195842743 0.009348732233047485 0.015106694772839546 -0.029654089361429214 -0.019218746572732925 0.006114874500781298
-0.001912255072966218 9.546356741338968E-4 0.007724205031991005 -0.05694272741675377 0.00793687254190445 -0.021980663761496544
```

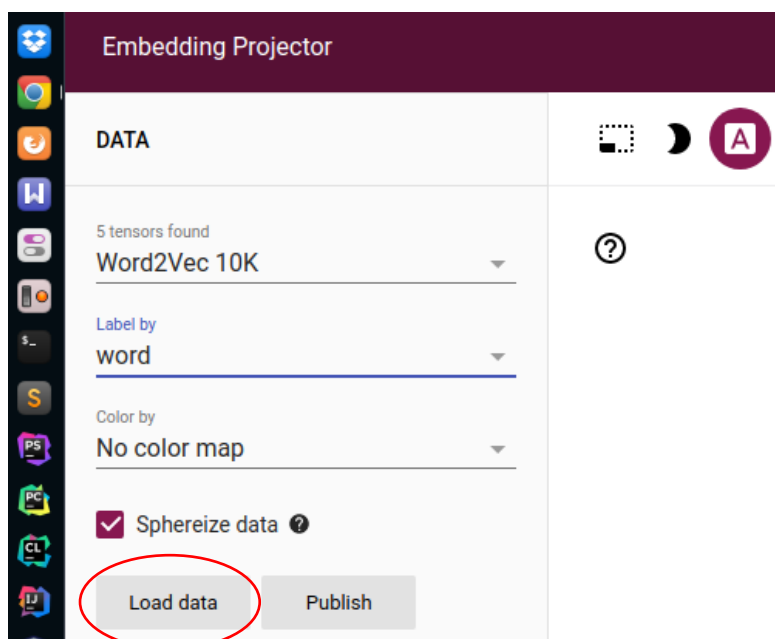
S'aprecia que cada paràgraf va seguit del seu vector de característiques. Això és el TSV.

## 8. Visualització a TensorFlow

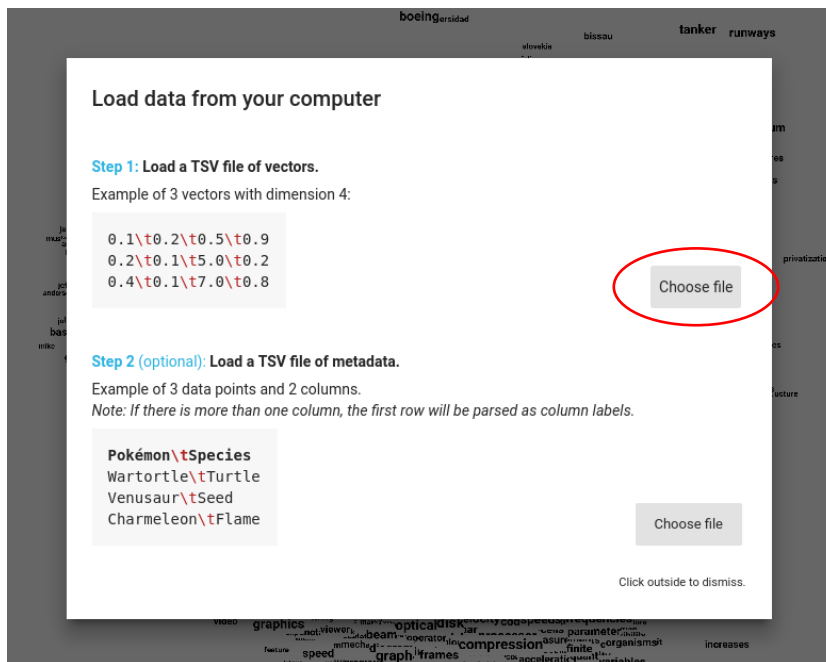
La darrera part de la pràctica consisteix en Visualitzar l'embedding amb l'eina TensorBoard de la suite tensorflow de google. Per fer-ho ga calgut adaptar al format requerit per tensorboard l'output de la fase anterior de la pràctica i carregar-lo al visualitzador de tensorboard.

El primer que cal fer és accedir a la web 'projector.tensorflow.org'. És una aplicació molt intuïtiva la qual només entrar ja ofereix al visitant una visualització de mostra. Concretament un word-count o comptador de paraules. Vegem però, què cal fer per carregar el nostre fitxer '.tsv' generat a l'apartat anterior per tal de visualitzar les distàncies entre els paràgrafs del Quijote.

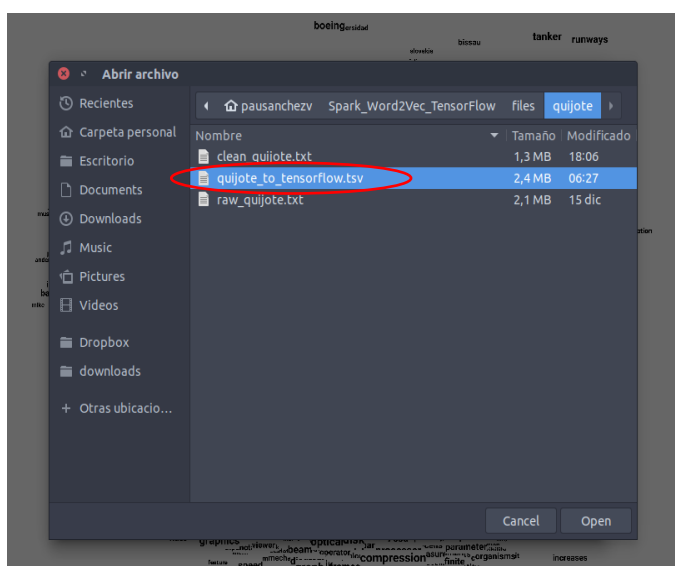
Una vegada dins la web, a la part esquerra trobem un botó 'Load Data', sobre el qual fem clic:



Aquest botó obre una finestra que permet dos tipus d'importació: el primer permet carregar un fitxer '.tsv' de vectors, que és justament el que hem generat a l'apartat anterior amb Spark. La segona opció, que no farem servir, permet importar fitxers de metadata.

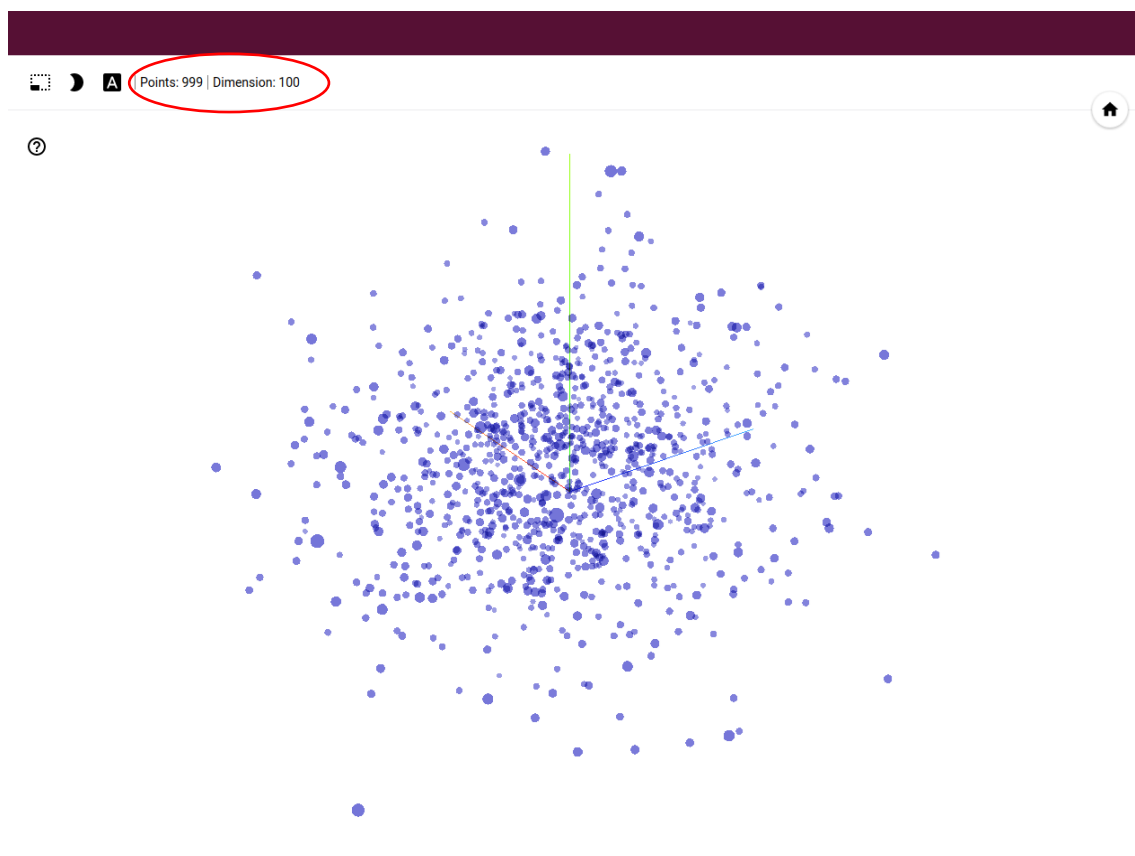


Seleccionem el fitxer '.tsv' que ha quedat gravat a la següent ruta relativa al projecte: `./files/quijote/quijote_to_tensorflow.tsv`, i l'obrim.



El primer que ofereix TensorFlow és una visualització de nodes on cada node de la visualització representa un paràgraf del quijote, en el cas que ens ocupa. Cal recordar com s'ha explicat prèviament que per millorar la visualització s'han agafat només els 1000

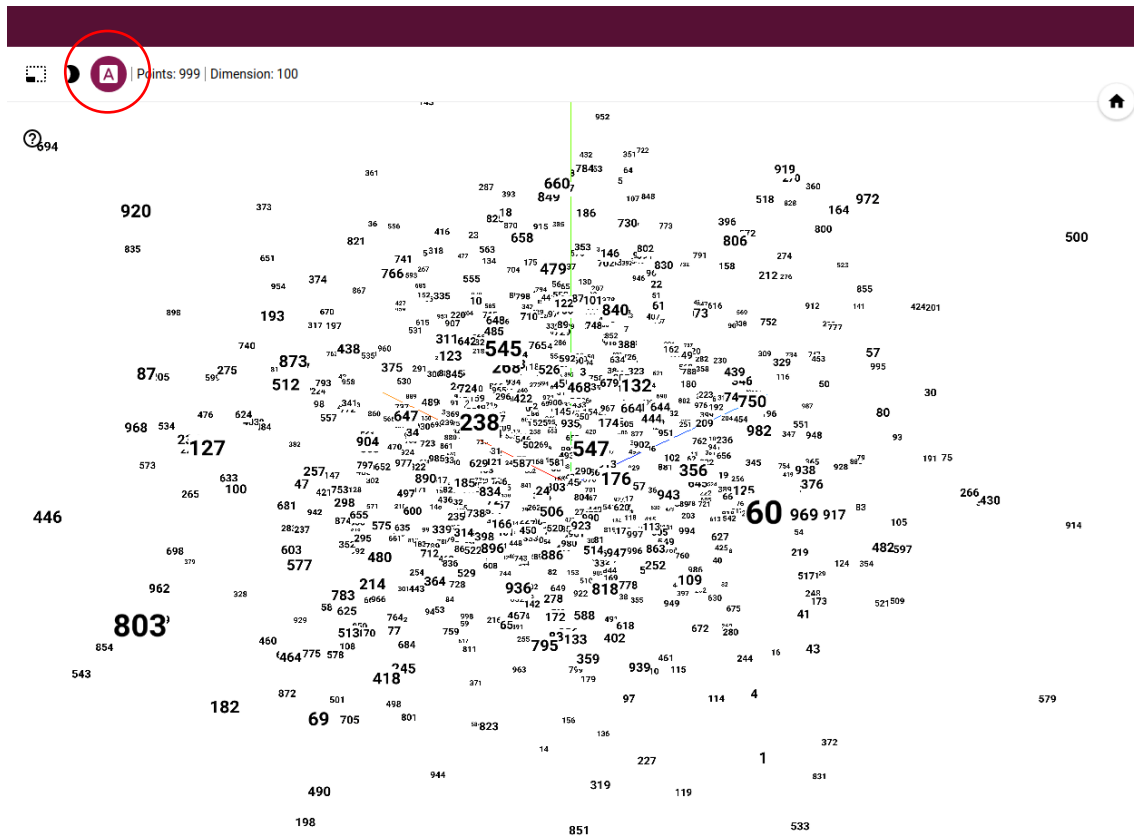
primers paràgrafs del llibre. Llavors els nodes esperats haurien de ser 1000 nodes amb dimensió 100 cadascun d'ells, ja que cada paràgraf està representat amb un vector de 100 característiques. A la següent visualització el lector pot comprovar que això és correcte si mira a la part superior on indica que hi ha 1000 nodes, de dimensió 100.



El problema de la visualització de nodes rau en el fet que l'usuari no sap a quin paràgraf correspon cada node. Per millorar la visualització cal prémer el botó de canvi de nodes,



aleshores l'usuari pot saber quin numero de paràgraf s'assembla més a quin, i acabar corroborant la similitud mirant els números de línia al fitxer de text.



A la imatge anterior s'aprecia que, per exemple, el paràgraf 60 del 803 estan molt lluny l'un de l'altre, distància que indica que els paràgrafs contenen paraules molt diferents, és a dir, que no s'assemblen. Per contra el lector pot fixar-se, per exemple, en el paràgraf número 100 i 633, estan molt a la vora, cosa que indica similitud entre ells!

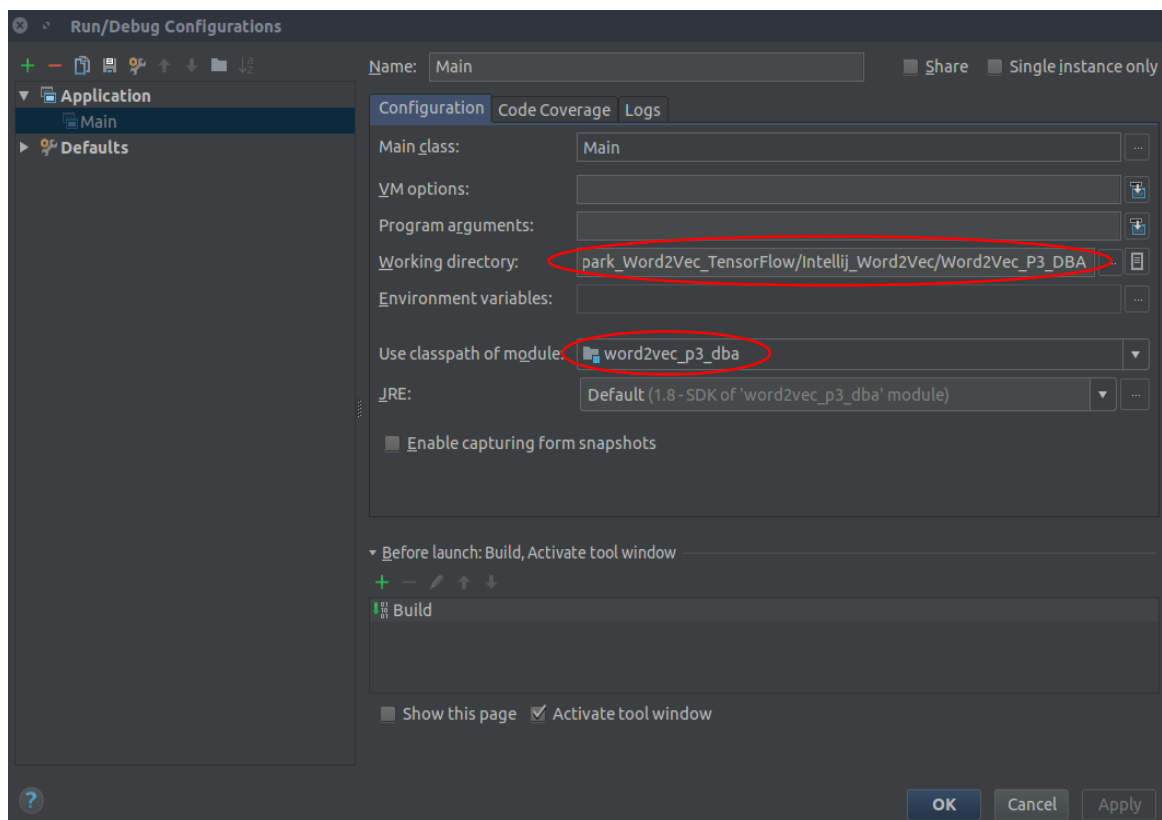
Enlloc del nombre de paràgraf es podria haver tret per pantalla el text del paràgraf sencer, però no s'ha cregut convenient fer-ho ja que la visualització és molt dolenta, solapant-se uns textos amb uns altres i acabant no entenent-se res del que s'està visualitzant. De manera que s'ha considerat mostra la visualització neta amb el número de paràgraf com a representant del mateix.

## Annex 1. Configuració de IntelliJ Idea

Una vegada acabada l'explicació de la pràctica cal mostrar la configuració que cal fer a IntelliJ Idea per tal de poder executar el projecte, en cas que el lector vulgui regenerar el fitxer '.tsv' per comprovar que funciona.

El projecte ja està configurat, de manera que n'hi hauria d'haver prou executant-lo directament ja que la configuració és a nivell de projecte i no d'IDE, no obstant a continuació s'expliquen uns petits trucs per si l'execució es negués a funcionar de primeres.

La pestanya 'Run/Debug Configurations' ha d'estar configurada de la següent manera, especialment els apartats 'Working Directory' i 'Use class path of module':



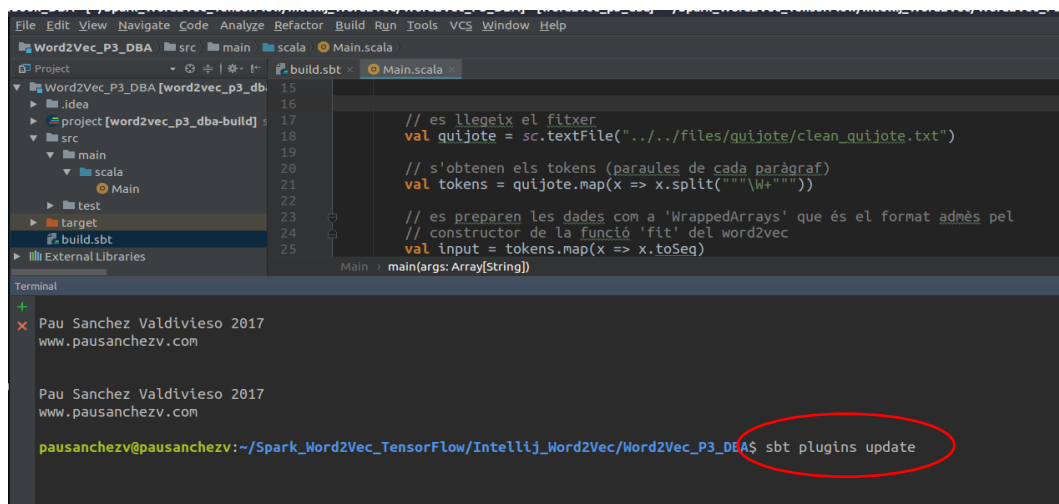
L'arxiu build.sbt és qui fa la importació i adaptació de les llibreries d'Spark a IntelliJ per a ser utilitzades amb Scala. Ha de tenir la següent configuració (després de modificar-lo cal refrescar el projecte per aplicar els canvis):

```
build.sbt was changed
1  name := "notebook_DBA"
2
3  version := "1.0"
4
5  scalaVersion := "2.11.8"
6
7  libraryDependencies += Seq(
8    "org.apache.spark" %% "spark-core" % "2.1.0",
9    "org.apache.spark" %% "spark-sql" % "2.1.0",
10   "org.apache.spark" %% "spark-mllib" % "2.1.0"
11 )
12
13
```

Ara cal fer un últim pas per actualitzar els plugins d'SBT:

- Clicant alt + F12 obrim una consola de linux dins els mateix IDE i redireccionat a la carpeta on es troba el projecte d'IntelliJ. Allà mateix escrivim la següent instrucció i cliquem intro:

>> sbt plugins Update



```

terminal
+ Pau Sanchez Valdivieso 2017
x www.pausanchezv.com

pausanchezv@pausanchezv:~/Spark_Word2Vec_TensorFlow/IntelliJ_Word2Vec/Word2Vec_P3_DBA$ sbt plugins update
[info] Loading settings from plugins.sbt ...
[info] Loading project definition from /home/pausanchezv/Spark_Word2Vec_TensorFlow/IntelliJ_Word2Vec/Word2Vec_P3_DBA/project
[info] Loading settings from build.sbt ...
[info] Set current project to notebook_DBA (in build file:/home/pausanchezv/Spark_Word2Vec_TensorFlow/IntelliJ_Word2Vec/Word2Vec_P3_DBA/)
In file:/home/pausanchezv/Spark_Word2Vec_TensorFlow/IntelliJ_Word2Vec/Word2Vec_P3_DBA/
sbt.plugins.IvyPlugin: enabled in word2vec_p3_dba
sbt.plugins.JvmPlugin: enabled in word2vec_p3_dba
sbt.plugins.CorePlugin: enabled in word2vec_p3_dba
sbt.plugins.JUnitXmlReportPlugin: enabled in word2vec_p3_dba
sbt.plugins.Giter8TemplatePlugin: enabled in word2vec_p3_dba
[info] Updating {file:/home/pausanchezv/Spark_Word2Vec_TensorFlow/IntelliJ_Word2Vec/Word2Vec_P3_DBA/}word2vec_p3_dba...
[info] Done updating.
[warn] Found version conflict(s) in library dependencies; some are suspected to be binary incompatible:
[warn] * commons-net:commons-net:2.2 is selected over 3.1
[warn] +- org.apache.spark:spark-core_2.11:2.1.0 (depends on 2.2)
[warn] +- org.apache.hadoop:hadoop-common:2.2.0 (depends on 3.1)
[warn] * com.google.guava:guava:14.0.1 is selected over 11.0.2
[warn] +- org.apache.curator:curator-recipes:2.4.0 (depends on 14.0.1)
[warn] +- org.apache.curator:curator-client:2.4.0 (depends on 14.0.1)
[warn] +- org.apache.curator:curator-framework:2.4.0 (depends on 14.0.1)
[warn] +- org.apache.hadoop:hadoop-hdfs:2.2.0 (depends on 11.0.2)
[warn] +- org.apache.hadoop:hadoop-common:2.2.0 (depends on 11.0.2)
[warn] Run 'evicted' to see detailed eviction warnings
[success] Total time: 7 s, completed 22-dic-2017 12:38:26
pausanchezv@pausanchezv:~/Spark_Word2Vec_TensorFlow/IntelliJ_Word2Vec/Word2Vec_P3_DBA$

```

Ara de ben segur que el projecte està apunt per fer clic sobre el botó 'play' i executar-lo sense problemes. Comprovem-ho!

```

92
93 //println(stringToTensorflowFile)
94
95
96 new PrintWriter("../files/quijote/quijote_to_tensorflow.tsv") {
97   write(stringToTensorflowFile)
98   close()
99 }
100
Main -> main(args: Array[String])

Run - Main
17/12/22 12:40:57 INFO Executor: Starting executor ID driver on host localhost
17/12/22 12:40:57 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on host localhost
17/12/22 12:40:57 INFO NettyBlockTransferService: Server created on 192.168.1.131:33157
17/12/22 12:40:57 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
17/12/22 12:40:57 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 192.168.1.131, 33157)
17/12/22 12:40:57 INFO BlockManagerMasterEndpoint: Registering block manager 192.168.1.131:33157 with 1944.0 MB memory
17/12/22 12:40:57 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 192.168.1.131, 33157)
17/12/22 12:40:57 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 192.168.1.131, 33157)
17/12/22 12:40:59 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
17/12/22 12:40:59 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS

Process finished with exit code 0

```

Es comprova a la imatge anterior que el projecte ha executat correctament amb la frase 'Process finished with code 0' (Els missatges en vermell és informació purament d'Spark, no són pas errors).

---

Autors: **Pau Sanchez i Guillem Rabionet**

Sota la tutela de: **Enric Biosca**

Universitat de Barcelona, semestre de tardor

4 de gener de 2018

---