UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Treball Final de Grau

Visual analysis of text similarity metrics

Autor: Adrián Martínez Montes
Director: Pere Pau Vázquez Alcocer

Grau d'Enginyeria Informàtica
Especialització de Computació
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Abstract (Català)

Aquest projecte compara diferents mètriques de distància aplicades a *embeddings* de documents sobre diferents gràfics conceptuals.

Al saber quines mètriques són efectives d'entre totes les analitzades sota uns límits establerts, facilita l'aplicació d'aquestes a altres àmbits. Per tant, coneixent els resultats del projecte tindrem la certesa que les mètriques funcionaran correctament, estalviant temps i optimitzant els recursos de qualsevol negoci quan es necessita una comparació de diferents documents per a una aplicació més professional.

Aquest projecte aplica dos models *NLP*: *Doc2Vec* i *Word2Vec*. L'objectiu d'aquesta investigació és profunditzar en el funcionament de les mètriques estudiades sobre aquests models. S'han creat diverses *pipelines* per automatitzar tot el processament de text, l'entrenament de models i l'exportació de resultats. Això ha permès poder crear tres contexts per model, fent que els resultats siguin més diversos i tindre més marge per comparar els resultats.

Entre tota l'experimentació realitzada, cal destacar l'estructura de proves combinatòries que ens ha permès crear resultats d'experimentació prou grans per a avaluar discriminant per Model-Context-Mètrica.

Finalment, les conclusions extretes han sigut bastant clares, la mètrica amb millor comportament general ha sigut *Cosine Similarity*, on per grups discriminats (Blogs, Científic, Wikipedia, Doc2Vec i Word2Vec) ha tingut el millor rendiment exceptuant Wikipedia, que ha estat en segon lloc.

Abstract (Castellano)

Este proyecto compara diferentes métricas de distancia aplicadas a embeddings de documentos sobre diferentes gráficos conceptuales.

Al conocer qué métricas son efectivas entre todas las analizadas bajo unos límites establecidos, facilita la aplicación de estas en otros ámbitos. Por lo tanto, conociendo los resultados del proyecto tendremos la certeza de que las métricas funcionarán correctamente, ahorrando tiempo y optimizando los recursos de cualquier negocio cuando se necesita una comparación de diferentes documentos para una aplicación más profesional.

Este proyecto aplica dos modelos NLP: *Doc2Vec* y *Word2Vec*. El objetivo de esta investigación es profundizar en el funcionamiento de las métricas estudiadas sobre estos modelos. Se han creado diversas *pipelines* para automatizar todo el procesamiento de texto, el entrenamiento de modelos y la exportación de resultados. Esto ha permitido poder crear tres contextos por modelo, haciendo que los resultados sean más diversos y tener más margen para comparar los resultados.

Entre toda la experimentación realizada, cabe destacar la estructura de pruebas combinatorias que nos ha permitido crear resultados de experimentación lo suficientemente grandes cómo para evaluar discriminando por Modelo-Contexto-Métrica.

Finalmente, las conclusiones extraídas han sido bastante claras, la métrica con mejor comportamiento general ha sido *Cosine Similarity*, dónde por grupos discriminados (Blogs, Científico, Wikipedia, Doc2Vec y Word2Vec) ha tenido el mejor rendimiento exceptuando Wikipedia, que ha quedado en segundo lugar.

Abstract (English)

This project compares different distance metrics applied to document embeddings above different conceptual graphics.

By knowing which metrics are effective among all the ones analysed under established limits, it facilitates the application of the aforementioned in other scopes. Therefore we have the certainty that the metrics will work properly, saving time and optimising the resources of any business when a comparison of different documents is needed for a more professional application.

This project applies two NLP models: Doc2Vec and Word2Vec. The goal of this research is to delve into how the studied metrics work on these models: different pipelines have been created in order to have all the text processes automated, the fitting of the model process and the result exportation. This has allowed us to create three contexts per model, making the results more diverse and having a bigger margin to compare the results.

Among all the experimentation done, the combinatory test structure should be highlighted as it has allowed us to create experimentation results large enough to evaluate the discrimination by Model-Context-Metric.

Finally, the conclusions extracted are clear: the metric with best general behaviour has been Cosine Similarity, where by discriminated groups (Blogs, Scientific, Wikipedia, Doc2Vec and Word2Vec) has had the best performance excepting the Wikipedia context, that has resulted in a second place.

Índex

Llista de Figures

Llista de Taules

1. Estructura del projecte

L'estructura d'aquesta memòria consta de 13 blocs com bé indica l'índex, en aquest apartat s'explicarà molt breument com es distribueix tota la informació de la memòria:

- **Contextualització i abast del projecte**: introducció i motivació del projecte.
- **Planificació**: com s'ha estructurat el projecte en tasques i com s'han assignat a una línia temporal.
- **Recursos i costos**: els recursos emprats i una estimació dels costos del projecte.
- **Estat de l'art**: breu introducció als actuals models de Processament Natural del Llenguatge
- **Conceptes**: explicacions molt bàsiques de diversos elements que es fan servir al llarg de tot el projecte.
- **Preprocessament de text i anàlisi**: obtenció dels datasets, com s'ha fet la pipeline de preprocessament i l'anàlisi d'aquests datasets.
- **Entrenament dels models**: com relacionem tota la teoría matemàtica dels models amb les llibreries que hem fet servir i l'utilització d'aquests.
- **Aplicació de mètriques**: com s'ha fet la rutina principal per tal de fer execucions massives d'aquesta i com s'exporten els diferents resultats per ta de fer-los servir més endavant.
- **Visualització de resultats**: els diferents tipus de gràfiques per tal de visualitzar les exportacions que s'han anat fent.
- **Escalabilitat del projecte**: s'especifica com es pot escalar el projecte per tal d'afegir nous elements com contextos o mètriques.
- **Experimentació**: és la secció on s'expliquen quins experiments s'han fet, com s'han fet, per què i els resultats.
- **Conclusions**: valoració general dels resultats de l'experimentació.
- **Bibliografia**: totes les fonts que s'han citat durant el projecte.
- **Apèndix**: scripts, resultats numèrics d'alguns experiments, etc.

2. Contextualització i abast del projecte

A aquesta secció contextualitzarem el projecte, donarem una visió global de les motivacions del projecte, els procediments d'una manera molt resumida i les eines que s'han emprat.

2.1. Títol / Objecte

Efectivitat de mètriques de similaritat entre diferents *embeddings* de documents i projecció sobre diferents gràfics conceptuals.

2.2. Justificació i utilitat

La utilitat de mesurar l'efectivitat de diferents mètriques sobre *embeddings* és poder avaluar quin mètode resulta més efectiu a l'hora de comparar documents. Sabent que un mètode és més efectiu que altre, es pot fer una avaluació dels costos de cada *embedding*.

2.3. Abast

Hem fet servir diferents tipus de generadors de models d*'embeddings*. Amb això hem pogut crear embeddings de diferents documents i hem pogut comparar aquests, una vegada tenint les diferències hem pogut fer comparacions entre grups de documents dos a dos, etc. Amb això s'ha fet estadística i s'ha arribat a diverses conclusions.

2.4. Especificacions bàsiques

Projecte amb Python mitjançant scripts per la part de *Natural Language Processing*, scripts amb *Jupyter Notebook* per avaluació de resultats massius i projecte angular amb *D3* per fer algunes visualitzacions.

2.4.1. Llibreries principals:

- **Gensim**: es fa servir per l'entrenament dels models i manipular aquests[1].
- **Nltk**: es fa servir per preprocessar els documents del projecte[2].
- **Scipy**: inclou eines per calcular mètriques.[3]
- **Matplotlib**: generació de gràfiques.[4]
- **Numpy**: manipulació d'estructures numèriques.[5]
- **Inflect**: es fa servir al preprocessament per les inflexions i transformació de números a numerals.[6]
- **Tika**: per extreure en format ascii el contingut d'un *pdf*.[7]
- **D3.js**: generació de gràfiques amb *javascript*.[8]

2.4.2. Generadors de models a fer servir

- Doc2Vec
- Word2Vec

2.4.3. Contextos a utilitzar

- Wikipedia
- Documents científics
- Entrades de blogs

[1] "Gensim: Topic modelling for humans - Radim Řehůřek." 22 dic. 2021, https://radimrehurek.com/gensim/. Es va consultar el 14 abr. 2022.
[2] "NLTK :: Natural Language Toolkit." https://www.nltk.org/. Es va consultar el 14 abr. 2022.
[3] "Scipy.org." 5 feb. 2022, https://scipy.org/. Es va consultar el 14 abr. 2022.
[4] "Matplotlib — Visualization with Python." https://matplotlib.org/. Es va consultar el 14 abr. 2022.
[5] "NumPy.org." https://numpy.org/. Es va consultar el 14 abr. 2022.
[6] "Welcome to inflect documentation! — inflect 5.5.1.dev0+gee50d4a ...." https://inflect.readthedocs.io/. Es va consultar el 14 abr. 2022.
[7] "chrismattmann/tika-python - GitHub." https://github.com/chrismattmann/tika-python. Es va consultar el 14 abr. 2022.
[8] "D3.js - Data-Driven Documents." https://d3js.org/. Es va consultar el 14 abr. 2022.

2.4.4. Mètriques de distàncies a utilitzar

- Euclídea
- Cosine similarity
- Word mover distance
- Manhattan
- Minkwoski
- Chebyshev
- Haversine

2.4.5. Mètriques no útils, però possibles candidates a evaluar

Es necessita un aplicatiu que transformi de *float* al format que faci servir cadascuna:

- Hamming
- Jaccard
- Sorensen-Dice

2.4.6. Reducció de dimensionalitats

- t-sne
- umap

2.4.7. Classificació

- tfidf

2.5. Desenvolupament del projecte - Alternatives

Tenim tres possibles escenaris problemàtics a aquest projecte:

- No tindre suficients models
- No tindre suficients contextos
- Resultats finals inconclusos

En cas de no haber trobat correlacions entre grups de documents als mateixos contextos, hauriem optat per altre dataset o bé optar per un altre context, o fins i tot fer servir altres mètriques.

2.6. Desenvolupament del projecte - Detall

Es compon de diversos passos per poder-ho dur a terme.

2.6.1. Obtenció de datasets

S'han buscat dos *datasets* per crear dos contextos i tenir textos de prova, un tercer ha sigut ja els models preentrenats per tindre un context més general.

2.6.2. Creació de la pipeline

La pipeline de preprocessament de text s'ha creat per tindre els datasets preparats per poder fer-los servir per a l'entrenament del model i fer-los servir com documents pels experiments.

2.6.3. Entrenament dels models

Per cada *dataset*, s'ha donat com a entrada cada document per entrenar el model.

2.6.4. Aplicació de mètriques

A aquest pas s'ha aplicat cadascuna de les mètriques amb tots els documents als diferents models disgregant per context, hem obtingut unes dades que exportarem a fitxers *csv* i *json*.

2.6.5. Visualització i conclusions

Els models entrenats s'han guardat a disc per tal de poder fer els diferents experiments sobre aquests, també s'han fet diverses anàlisis d'aquests per veure com es comporten.

S'han representat els resultats i extretes unes conclusions, aquestes no han sigut inconcluses, ja que s'ha anat més enllà intentant acurar més els experiments.

2.7. Normativa i el seu compliment

No operem amb dades personals de ningú. Totes les dades recollides són de lliure ús, wikipedia, documents científics i entrades de blogs públics.

3. Planificació

A aquesta secció veurem com hem planificat el projecte, la descomposició en tasques petites i com les hem assignat en el temps. Hem fet també un desenvolupament de les dependències entre les tasques per detectar tasques que bloquegen altres. Finalment, s'han valorat plans alternatius en cas que sorgeixin imprevistos.

En aquest projecte s'ha treballat una mitja de 3 hores diàries, la duració d'aquest projecte ha sigut d'unes 230 hores sumades a totes les tasques sense sumar el treball extra que s'ha derivat dels problemes que han sorgit.

3.1. Tasques

A continuació veurem les tasques que s'han dut a terme durant el projecte, s'ha dividit en tasques que es poden dur a terme entre un dia i diversos dies, per exemple, una tasca que ocupi 16 hores es durà a terme en tres o quatre dies, al diagrama de Gantt es detallen les duracions en hores.

T1: Introducció a gensim: Aprendre com es fan servir els generadors de models Word2Vec i Doc2Vec.

T2: Preparació del projecte de python: Crear l'estructura de carpetes, els fitxers bàsics i la instal·lació de paquets *pip3*.

T3: Preparació del primer dataset: El primer dataset consisteix en una pila de documents científics en pdf. Amb python generarem un script per recollir la informació d'aquests en format text i guardar-ho pel posterior ús.

T4: Creació de la pipeline de processament de text: Aquesta pipeline ens permet introduir un text i tornar-ho preprocessat amb totes les regles que se solen fer servir.

T5: Creació del primer context en Doc2Vec: Entrenar el model amb tots els documents científics preprocessats i deixar preparats tots els *getters* per tindre fàcil accés a aquest.

T6: Similaritat de dos documents amb Doc2Vec: Script que mitjançant dos documents diferents, afegir paraules d'un document a l'altre, obtenir els *embeddings* d'aquests dos documents i calcular el *cosine similarity* entre els dos documents. Amb això hem pogut confirmar que estem fent servir correctament el model generat.

T7: Creació del primer context a Word2Vec: Entrenar el model *Word2Vec* amb els documents científics preprocessats.

T8: Similaritat de dos documents a Word2Vec: Mateix funcionament que amb Doc2Vec. La primera prova per comprovar de que ho estem fent correctament.

T9: Visualització dels models generats: Generar diversos gràfics per veure com s'estan comportant els models amb els diferents inputs. Per exemple, comprovar quines paraules haurien d'estar properes i quines ho estan. Agafar diferents documents, crear els embeddings d'aquests i veure que propers són depenent de com similars són.

T10: Creació d'un segon context específic: Això ja inclou la preparació d'aquest dataset i l'entrenament dels models per Doc2Vec i Word2Vec.

T11: Creació d'un tercer context més general: La idea és que les nostres proves puguin estar contingudes en cadascun dels contextos i comparar els resultats, tindre un context general ens dóna llum quan hem fet proves mixtes (amb documents dels dos datasets específics).

T12: Preparació del projecte d'angular: Necessitarem un entorn de visualització de dades on integrarem la llibreria D3.

T13: Creació del núvol de punts en D3: Ens ha sigut útil per representar les dades sobre un gràfic en 2 dimensions mitjançant fitxers de tipus *CSV*.

T14: Creació de matriu de similaritat en D3: En general, per comparar un a un documents o per classes, la idea és poder fer grups de documents i comparar-los dos a dos mitjançant fitxers de tipus *JSON* (Json és el més pràctic per representar matrius).

T15: Classificació tfidf genèrica: Per agilitzar (o inclús possibilitar) alguns càlculs de mètriques, com per exemple *WMD*, que té un cost computacional elevat; el que hem fet és crear una classificació *tfidf* genèrica amb input de tots els documents que tenim i els documents a classificar, així hem acotat el nombre de paraules per document seleccionat a només les més rellevants.

T16: Agrupació genèrica per classes: En vista que les comparacions un a un són tedioses, poder agrupar per classes ens ha facilitat molt el treball, per exemple, documents de dos contexts diferents, una classe per context, calculem les mètriques i fem la mitja de les distàncies obtingudes entre els elements de la classe, així tindrem com a resultat l'aplicació de la mètrica dins d'aquesta classe.

T17: Mètrica 1: Euclidean:
Tota mètrica té les següents proves:
Per cada Model:

- Per cada Context:
    - Comparar dos a dos n documents (n/2 documents pel primer context específic i n/2 del segon context específic) i generar json amb matriu de similaritat.
    - Comparar dos a dos fent servir agrupació genèrica per classes, en tots els casos tindrem 4 quadrants, dos d'ells idèntics i els altres dos amb els resultats de les comparacions internes de cada classe, generar *json* amb la mateixa matriu de similaritat.

- Amb *tfidf* seleccionar les paraules més importants de tots els documents i fer una comparació que abasti més documents, classificar per classes i visualitzar en quatre quadrants.

T18: Mètrica 2: Cosine Similarity: Ídem T17
T19: Mètrica 3: Word Mover Distance: Ídem T17
T20: Mètrica 4: Manhattan: Ídem T17
T21: Mètrica 5: Minkwoski: Ídem T17
T22: Mètrica 6: Chebyshev: Ídem T17
T23: Mètrica 7: Haversine: Ídem T17

T24: Obtenir les gràfiques de cada matriu i documentarles: S'han deixat preparats dos generadors de gràfics i solament hem tingut que anar passant els arxius *JSON* generats a cada mètrica i anar extraient els resultats.

T25: Experimentació i conclusions: Aquesta tasca a aquest tipus de projecte ha tingut un abast de bastants hores, ja que tot l'altre sense això no té cap sentit.

T26: Redacció del document: Redactar tots els detalls del projecte, tots els problemes que han anat sorgint i incloure tota la documentació addicional.

O1: (Opcional) Proves addicionals: Proves addicionals que han anat succeint, aquí és el moment de realitzar-les.

O2: (Opcional) Conclusions proves addicionals: Ídem O1.

3.2. Dependències de les tasques

S'han agrupat les dependències d'acord amb un graf dirigit acíclic.

T26 > T25

T25 > T24, T23, T22, T21, T20, T19, T18, T17

T24 > T23, T22, T21, T20, T19, T18, T17, T16, T14, T13

T23, T22, T21, T20, T19, T18, T17 > T16, T15, T11, T10, T7

T16, T15 > T2

T14, T13 > T12

T12 > No té

T11, T10 > T4, T1

T9 > T7, T5

T8 > T7

T7 > T4, T3, T1

T6 > T5

T5 > T4, T3, T1

T4 > T2

T3 > T2

T2 > No té

T1 > No té

3.3. Planificació horària

gantter™

Project Name **tfg**

		Nombre	Duración	Inicio	Fin	Predecesoras	Recursos	Custom 1	Custom 2	Custom 3	Custom 4	Custom 5
1		T1	10horas	03/08/2021	03/09/2021							
2		T2	2horas	03/11/2021	03/11/2021							
3		T3	4horas	03/11/2021	03/11/2021	2						
4		T4	8horas	03/15/2021	03/15/2021	2						
5		T5	8horas	03/17/2021	03/17/2021	1,3,4						
6		T6	12horas	03/18/2021	03/19/2021	5						
7		T7	8horas	03/22/2021	03/22/2021	1,3,4						
8		T8	10horas	03/23/2021	03/24/2021	7						
9		T9	3horas	03/25/2021	03/25/2021	5,7						
10		T10	5horas	03/26/2021	03/26/2021	1,4						
11		T11	8horas	03/30/2021	03/30/2021	1,4						
12		T12	2horas	04/01/2021	04/01/2021							
13		T13	16horas	04/01/2021	04/05/2021	12						
14		T14	16horas	04/05/2021	04/06/2021	12						
15		T15	10horas	04/09/2021	04/12/2021	2						
16		T16	15horas	04/13/2021	04/14/2021	2						
17		T17	10horas	04/19/2021	04/20/2021	7,10,11,15,16						
18		T18	8horas	04/21/2021	04/21/2021	7,10,11,15,16						
19		T19	7horas	04/23/2021	04/23/2021	7,10,11,15,16						
20		T20	6horas	04/26/2021	04/26/2021	7,10,11,15,16						
21		T21	5horas	04/28/2021	04/28/2021	7,10,11,15,16						
22		T22	4horas	04/30/2021	04/30/2021	7,10,11,15,16						
23		T23	3horas	05/03/2021	05/03/2021	7,10,11,15,16						
24		T24	30horas	05/03/2021	05/07/2021	13,14,16,17,18,1						
25		T25	30horas	05/12/2021	05/17/2021	17,18,19,20,21,2						
26		T26	40horas	05/19/2021	05/25/2021	25						

PNG Generated On: 3/8/2021, 8:33:06 PM

Figura 1: Planificació horària de les tasques, generat pel software gantter..

3.4. Diagrama de gantt

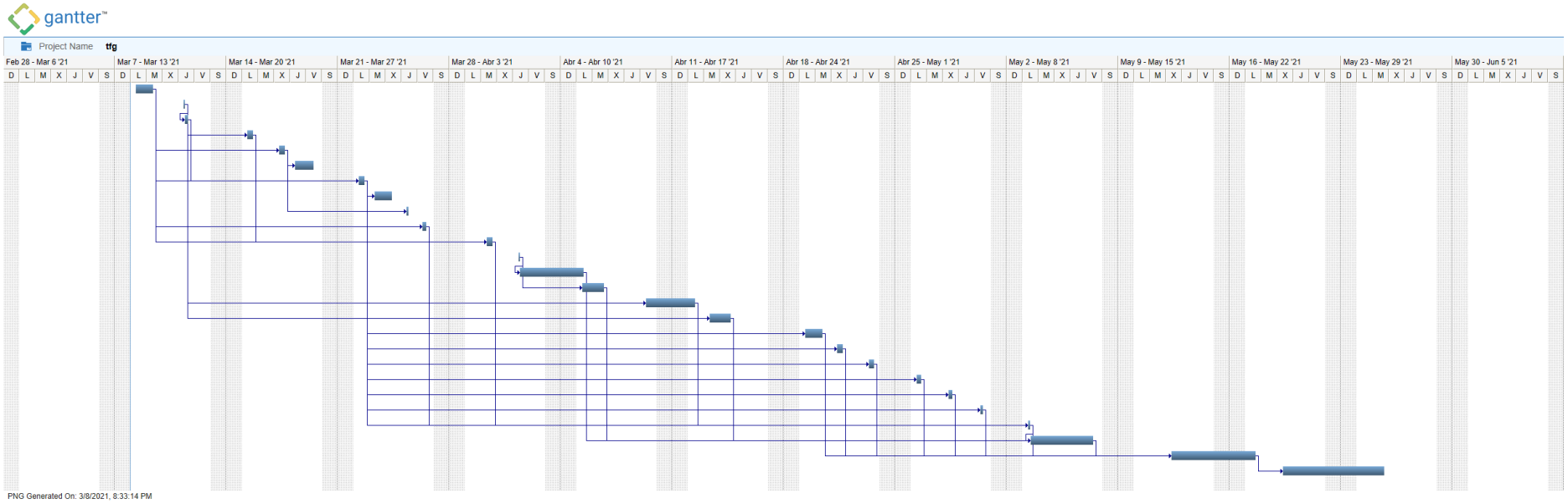

Figura 2: Diagrama de gantt horari, generat pel software gantter.

3.5. Gestió del projecte: Plans alternatius

Els riscos que identifiquem a aquest projecte són:

3.5.1. Risc 1: Pocs models a analitzar

Pot passar que tinguem resultats inconclusos, en aquest cas una de les solucions és afegir un altre model i passar per diferents tasques ja proposades anteriorment.
En aquest cas haurem d'entrenar un altre tipus de model, entrenar els diferents contextos, fer les proves amb totes les mètriques suggerides i redactar conclusions.

TR1.1: Creació dels tres contextos al nou model
TR1.2: Aplicar les 7 mètriques proposades
TR1.3: Conclusions

3.5.2. Risc 2: Pocs contextos emprats

En aquest cas és que els contextos fets servir són molt semblants a ulls dels models i pot ser que no tinguem resultats diferenciadors entre aquests.
La solució implica buscar altres *datasets* i entrenar de nou els models.

TR2.1: Buscar *dataset*
TR2.2: Entrenar models
TR2.3: Aplicar mètriques
TR2.4: Conclusions

3.5.3. Risc 3: Visualització d'estadístiques poc clares

Pot succeir que les visualitzacions que estem fent servir no siguin les més adients i hàgim de recórrer a altres gràfiques amb altres fons de dades.
En aquest cas haurem de retocar les sortides de dades de cada mètrica per ajustar-les als requisits d'entrada als gràfics.

TR3.1: Creació de la nova visualització
TR3.2: Ajust de la sortida de dades de les diferents mètriques
TR3.3: Conclusions

4. Recursos i costos

A aquesta secció veurem els recursos i l'estimació dels costos del projecte, veurem com els recursos humans són molt més rellevants que els materials. Farem una descomposició dels costos en taules i una valoració de la viabilitat econòmica del projecte.

4.1. Recursos

Aquest apartat explica els diferents càlculs que s'han fet a l'hora de calcular tots els recursos que s'han requerit per fer el projecte.

4.1.1. Recursos humans

Existeixen dues persones considerats com a recursos humans en aquest projecte:

Director del projecte: S'encarregarà de dirigir el projecte, posa les pautes del projecte.
Programador: S'encarregarà de fer tota la implementació de tasques.

4.1.2. Recursos materials

Els recursos materials considerats han sigut tots els recursos als que ha tingut abast l'alumne que s'ha encarregat de desenvolupar el projecte.

Equip amb connexió a internet
- Ordinador
- Connexió a internet
- Teclat
- Ratolí
- Pantalla

Eines de desenvolupament
- *PyCharm* per la part d'entrenament de models.
- *WebStorm* per la part de visualització de dades.
- *Word2Vec*: un dels generadors de models.
- *Doc2Vec*: l'altre generador de models.
- *Jupyter Notebook*: creació i execució *d'scripts* parcials

PyCharm, Word2vec, Doc2vec i *Jupyter Notebook* són d'ús lliure.

Eines de desplegament
- *Github* (gratuït per estudiants)
- Navegador web (gratuït)

4.2. Estimació dels costos

A l'estimació dels costos es detallen els preus de cada recurs humà, de cada cost directe i de cada cost passiu; seguidament es fa un cost global i es donen els detalls dels costos per impediments.

4.2.1. Recursos humans

Els costos humans base són els següents:

Cost base recursos humans bruts		
Director de projecte	20€	h
Programador	12€	h

Taula 1: cost base recursos humans bruts.

Calculem el cost aplicant la seguretat social:

Cost base recursos humans nets		
Director de projecte	26€	h
Programador	16€	h

Taula 2: cost base recursos humans nets.

A partir d'ara tots els càlculs es faran amb el cost net.

4.2.1.1. Costos recursos humans

El càlcul per tasques són els següents:

Nom de la tasca	Dies	Hores diàries	Recurs	Cost
T1: Introducció a gensim	2	4	D, P	333€
T2: Preparació del projecte de python	0,5	4	P	31€
T3: Preparació del primer dataset	0,5	4	D, P	83€
T4: Creació de la pipeline de preprocessament de text	1	4	P	62€
T5: Creació del primer context amb doc2vec	2	4	P	125€
T6: Similaritat de dos documents amb doc2vec	3	4	D, P	499€
T7: Creació del primer context amb word2vec	1	4	P	62€
T8: Similaritat de dos documents amb word2vec	2	4	P	125€

T9: Visualització dels models generats	0,5	4	P	31€
T10: Creació d'un segon context específic	1	4	P	62€
T11: Creació d'un tercer context més general	2	4	P	125€
T12: Preparació del projecte d'angular	0,5	4	P	31€
T13: Creació de núvol de punts amb d3	4	4	P	250€
T14: Creació de matriu de similaritat amb d3	4	4	P	250€
T15: Classificació tfidf genèrica	3	4	D, P	499€
T16: Agrupació genèrica per classes	4	4	P	250€
T17: Mètrica 1: Euclidean	3	4	P	187€
T18: Mètrica 2: Cosine Similarity	2	4	P	125€
T19: Mètrica 3: Word Mover Distance	2	4	P	125€
T20: Mètrica 4: Manhattan	2	4	P	125€
T21: Mètrica 5: Minkwoski	1	4	P	62€
T22: Mètrica 6: Chebyshev	1	4	P	62€
T23: Mètrica 7: Haversine	1	4	P	62€
T24: Obtenir els gràfics de cada matriu i documentar-los	5	4	P	312€
T25: Conclusions	5	4	D, P	832€
T26: Redacció del document	6	4	P	374€
	59		Total	5.086€

Taula 3: càlcul de costos per tasca.

4.2.2. Costos directes

Considerarem jornades de 4 hores, parlem d'un 12,5% del període màxim de 8 anys.

Equip personal - Ordinador personal

El cost de l'ordinador personal ha sigut de 1000€,
Costos:

- Ordinador: 1000€
- Teclat: 20€
- Ratolí: 20€
- Pantalla LED: 150€

Càlcul:
((1000+20+20+150)*0,125*59*4)/(8*365) = 12,02€
(Coste total * 12,5% * 59 dies * 4 hores) / (8 anys / 365 dies)

4.2.3. Costos passius

Línia internet
Fibra 40€/mes càlcul de l'amortització
(480*59*4)/(8*365)= 39€
(Coste total anualitzat * 59 dies * 4 hores) / (8 anys / 365 dies)

Electricitat
Ordinador consumint electricitat a 350W una mica més elevada pel fet que fem operacions bastant grans a l'hora de fer els diferents càlculs computacionals.

Calculem cost d'entrenament de tots els models:
(350W * 24 hores * 6 dies) / 59 dies totals de projecte = 854W/h per dia = => d'entrenament dels models
(350W * 24 hores * 10 dies) / 59 dies totals de projecte = 1423W/h per dia => temps total que ha estat executant els experiments fent ús intensiu de la màquina.

Calculem Watts / hora dia
(350W * 4 hores al dia de treball) + (854W/h per dia + 1423W/h per dia) = 3677W/h per dia

Convertim a KiloWatts hores dia que és la referència per aplicar el preu
3677W/h / (1000W / 1 KW) = 3,677 KW hores/dia

Preu mitjà KW al dia: 0,3*3,677 = 1,1€ al dia
Preu al mes: 1,1*30 = 33€
Preu anual: 396€
Càlcul amortització:
(396*59*4)/(8*365)= 32€
(Cost total anualitzat * 59 dies * 4 hores) / (8 anys / 365 dies)

Eines de desenvolupament i desplegament
Webstorm té una anualitat de 129€ per any
(129*59*4)/(8*365)=10,43€
(Cost total anualitzat * 59 dies * 4 hores) / (8 anys / 365 dies)

Costs materials	Preu base		Anualitzat		Amortització
Equip personal					
Ordinador	1.000€				
Teclat	20€				
Ratolí	20€				
Pantalla LED	150€				12€
Total	1.190€				
Despeses passives					
Línia internet	40€	mes	480€		39€
Electricitat	33€	mes	396€		32€

Webstorm	129€	anual	129€		10€
				Total amort.	93€

Taula 4: costos passius.

4.2.4. Costos totals

Recurs	Cost
Recursos humans	5.086€
Recursos materials	93€
Total:	5.179€

Taula 5: costos totals.

Quasi tot el nostre cost és de capital humà, ja que el projecte es basa a una investigació software sense una infraestructura gran.

4.2.5. Costos per impediments

Hem fet el càlcul dels costos per impediments de manera que veiem els costos de cada risc per separat i del cost total de tots els riscos.

Impediments	Dies	Hores diàries	Recurs	Cost
Risc 1: Pocs models a analitzar				
TR1.1: Creació dels tres contextos al nou model	2	4	P	125€
TR1.2: Aplicar les 7 mètriques proposades	3	4	P	187€
TR1.3: Conclusions	3	4	P	187€
Risc 2: Pocs contextos emprats			Total	499€
TR2.1: Buscar dataset	1	4	P	62€
TR2.2: Entrenar models	1	4	P	62€
TR2.3: Aplicar mètriques	3	4	P	187€
TR2.4: Conclusions	3	4	P	187€
Riesgo 3: Visualització d'estadístiques poc clares			Total	499€
TR3.1: Creació de la nova visualització	2	4	P	125€
TR3.2: Ajustament de la sortida de dades de les diferents mètriques	1	4	P	62€
TR3.3: Conclusions	3	4	P	187€
			Total	374€
			Total al pitjor cas	1.373€

Taula 6: costos per impediments.

4.3. Viabilitat econòmica

Es tracta d'un projecte d'investigació software sense infraestructura gran, el qual solament es podria obtenir fons mitjançant alguna subvenció a ajut de l'estat; o per mitjà d'alguna empresa interessada que li pot ser útil l'anàlisi i conclusions del projecte.

4.3.1. Viabilitat per mitjà de subvenció o ajut de l'estat

S'hauria de sortir a concurs i intentar obtenir algun ingrés mitjançant aquest medi.

4.3.2. Viabilitat per mitjà d'ajudes d'empreses

En aquest cas s'haurien de buscar espai d'empreses on emprenedors, estudiants, investigadors, etc; presentin les seves idees, s'hauria de preparar una bona exposició per atreure l'atenció i recalcar les utilitats que pot tindre aquest projecte.

4.3.3. Viabilitat per mitjà de plataformes de micromecenatge

És complicat que usuaris tradicionals de les xarxes puguin cridar-los l'atenció d'aquest projecte, ja que no aporta un bé de consum immediat.

4.3.4. Impacte econòmic

L'allotjament web per poder mantenir les diferents vistes de les dades, depèn de l'allotjament, no necessita característiques molt elevades, amb un pla molt bàsic és suficient. Rondaria els 10€ al mes domini + allotjament.

5. Estat de l'art

En aquesta secció presentarem les actuals solucions que hi ha al mercat sense comptar les que fem servir al projecte, aquestes s'expliquen amb més detall a la secció corresponent.

Existeixen diversos models molt més sofisticats que els que tenim, els dos models que es fan servir al projecte són de 2013 (Word2Vec) i 2014 (Doc2Vec), els comentats a l'estat de l'art són molt més recents. El Processament de Llenguatge Natural avança molt ràpidament, ja que dóna solucions a múltiples entorns com medicina, traductors, assistència tècnica, etc; és per això que el desenvolupament ha sigut tan vertiginós en aquests últims anys, encara que l'entrenament d'aquests models encara té com a barrera de desenvolupament la capacitat dels equips informàtics, cada vegada es necessiten supercomputadors més grans per superar l'efectivitat dels anteriors models, està per veure si trobem arquitectures que no facin un ús tan intensiu de la maquinària i siguin tan acurats com els actuals.

5.1. BERT (2018)

El nom de BERT[9] es dóna per "Bidirectional Encoder Representations from Transformers". La innovació d'aquest model és que el seu entrenament és bidireccional, és a dir, té en compte paraules cap a la dreta i cap a l'esquerra. Fa servir el Transformer, un mecanisme que aprèn les relacions contextuals entre paraules a un text.

El transformer a la seva versió més simple, té dos grans elements, el *encoder* i el *decoder*, el encoder llegeix el text d'entrada tot de cop, això permet que el sentit de lectura no sigui ni esquerra-dreta ni dreta-esquerra (per això es diu també que és no direccional) i el decoder produeix una predicció per la tasca.

El Fine-Tuning és el core del BERT, consta de dos processos[10]:

Next Sentence Prediction (NSP)[11]
Consisteix a agafar parells de sentències com a inputs del model, alguns d'aquests parells són veritables i altres no, la idea és que el model digui si són parells o no i després fer feedforward amb les respostes correctes.

Masked LM (MLM)
A la implementació més simple del model el que fa és agafar el 15% de les paraules i substituir-los per un token anomenat MASK que representa una incògnita, l'objectiu és predir aquests tokens Masked, al final d'aquest procés la Loss Function de Bert té només en consideració la predicció dels

[9] "arXiv:1810.04805v2 [cs.CL] 24 May 2019." 24 may. 2019, https://arxiv.org/pdf/1810.04805. Es va consultar el 14 abr. 2022.
[10] "BERT Explained: State of the art language model for NLP." 10 nov. 2018, https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270. Es va consultar el 14 abr. 2022.
[11] "How to Fine-Tune BERT Transformer Python | Towards Data Science." 15 jun. 2021, https://towardsdatascience.com/how-to-train-bert-aaad00533168. Es va consultar el 14 abr. 2022.

tokens Masked i no dels altres. Això fa que el model convergeixi més lent que els models direccionals.

El MLM es fa servir amb sentències per entrenar el model i sentències del llenguatge ja introduïdes.

5.2. RoBERTa (2019)

RoBERTa[12] està construïda[13] sobre BERT, el que fa és iterar sobre el procés de pre-entrenament de BERT permetent entrenar el model durant més temps amb major paquets d'informació, eliminant la següent sentència a predir objectiu i dinàmicament canviant el patró de *masking* aplicat a les dades d'entrenament.

5.3. OpenAI's GPT-3 (2020)

El GPT-3[14] és un model basat en el Transformer, està dissenyat per generar text llegible per humans. Pot generar codi, històries, poemes, etc. El nombre de *trainable parameters* és de 175 Bilions, és el model que fins ara utilitza més, un *trainable parameter* és un pes o un bias de la xarxa neuronal que hi ha dins d'un model.

Donat un text d'entrada, el model pot determinar probabilísticament què tokens d'un vocabulari conegut vindrà després.

Hi ha diverses versions de GPT-3 més reduïdes, on els paràmetres són bastant més petits[15].

5.4. ALBERT (2020)

ALBERT[16] és un model basat en el BERT però pensat per fer-ho servir en màquines molt més lleugeres com ordinadors de casa. Amb alguns canvis a la seva arquitectura han fet que s'aconsegueixi tindre el mateix rendiment que el BERT però només amb una fracció de *trainable parameters*[17].

---

[12] "RoBERTa: An optimized method for pretraining self-supervised NLP ...." https://ai.facebook.com/blog/roberta-an-optimized-method-for-pretraining-self-supervised-nlp-systems/. Es va consultar el 14 abr. 2022.
[13] "pytorch/fairseq: Facebook AI Research Sequence-to ... - GitHub." https://github.com/pytorch/fairseq. Es va consultar el 14 abr. 2022.
[14] "OpenAI GPT-3: Everything You Need to Know - Springboard." 1 nov. 2021, https://www.springboard.com/blog/data-science/machine-learning-gpt-3-open-ai/. Es va consultar el 14 abr. 2022.
[15] "[2005.14165] Language Models are Few-Shot Learners - arXiv." 28 may. 2020, https://arxiv.org/abs/2005.14165. Es va consultar el 14 abr. 2022.
[16] "machine-learning-articles/albert-explained-a-lite-bert.md at main." 6 ene. 2021, https://github.com/christianversloot/machine-learning-articles/blob/main/albert-explained-a-lite-bert.md. Es va consultar el 14 abr. 2022.
[17] "ALBERT: A Lite BERT for Self-supervised Learning of Language ...." https://arxiv.org/abs/1909.11942. Es va consultar el 14 abr. 2022.

5.5. XLNet (2020)

Pren el BERT com a punt de partida i el millora[18], un dels grans inconvenients del BERT és que al procés de MLM els símbols que es fan servir com "[MASK]" són absents d'informació real en el moment del Fine-Tuning fent que s'introdueixi soroll[19].

XLNet funciona també amb el *Transformer*, és un model autoregressiu, fent que el soroll comentat no es pugui donar. Els models autoregressius són unidireccionals, però al fer servir el *Permutation Language Modeling* se supleix aquesta carència[20].

[18] "XLNet - Hugging Face." https://huggingface.co/docs/transformers/model_doc/xlnet. Es va consultar el 14 abr. 2022.
[19] "XLNet: Autoregressive Pre-Training for Language Understanding." 6 jul. 2020, https://towardsdatascience.com/xlnet-autoregressive-pre-training-for-language-understanding-7ea4e0649710. Es va consultar el 14 abr. 2022.
[20] "XLNet — SOTA pre-training method that outperforms BERT - Medium." 25 jun. 2019, https://medium.com/logits/xlnet-sota-pre-training-method-that-outperforms-bert-26d4e9978983. Es va consultar el 14 abr. 2022.

6. Conceptes

Hem repassat l'estat de l'art i hem vist cinc dels actuals models existents al mercat, ens queda per explicar els dos models que hem omès i que es fan servir en aquest projecte a més de diversos conceptes elementals.

En aquesta secció s'explicaran tots els conceptes relacionats amb el projecte i que són necessaris saber per poder entendre els posteriors experiments que s'han fet i les pipelines de processament.

Trobarem conceptes tan rellevants com *NLP*, *embedding*, reducció de la dimensionalitat, els contexts que s'han fet i les mètriques que es fan servir.

6.1. Bàsics

En aquesta secció tractarem els conceptes més bàsics, NLP i els Embeddings, aquests dos conceptes són clau a l'hora de desenvolupar el projecte.

6.1.1. Natural Language Processing (NLP)

És l'habilitat d'un programa de computador d'entendre el llenguatge humà parlat i escrit, no només això, ha de tindre la capacitat d'analitzar grans volums de dades per tal de poder executar altres tasques que depenguin del que s'ha interpretat.

És una subbranca de coneixement de la lingüística, ciències de la computació i intel·ligència artificial.

Els usos del NLP es poden estendre a qualsevol àmbit, medicina, negocis, assistents domèstics, etc.

Hi ha hagut diverses fases de desenvolupament, les primeres versions dels programes de computador de NLP, des del 1950 als anys 90, es feia simbòlicament, és a dir, s'emulava el llenguatge natural aplicant un conjunt de regles preestablertes, evidentment aquesta tècnica implicava un sobreesforç humà per tal d'assegurar les màximes regles possibles.

Més tard, des dels anys noranta cap al 2010 es van començar a fer servir models estadístics de *machine learning*, els computadors començaven a ser més potents i cada vegada era més viable executar programes de computador més intensius.

L'etapa actual de NLP es basa en xarxes neuronals, ja que són molt escalables i els computadors són exponencialment més potents que fa pocs anys, en aquest projecte es basa en aquest tipus de NLP.

6.1.2. Embedding

És la representació vectorial d'un element lingüístic a un model concret, és a dir, dins d'un model podem representar una paraula, frase, document, etc; en termes de vectors i amb aquests vectors obtinguts poder desenvolupar altres tasques, com per exemple, poder calcular automàticament com són de semblants dos documents i prendre altres decisions de negoci.

6.2. Representacions de dades

Les representacions de dades ens ajuda a codificar el vocabulari per tal que un model el pugui entendre i operar amb aquest.

6.2.1. One-hot encoding

Com bé el nom indica és una codificació d'un bit actiu, es tracta de fer un vector de bits de la mida del vocabulari, per exemple, si el vocabulari consta de 200 paraules, la mida del vector de bits és de 200 on cada posició del vector s'assigna a una paraula, i només un d'aquests bits podrà estar a 1 al mateix temps representant la paraula en qüestió dins de la codificació que ens hem inventat.

Exemple de *one hot encoding*:

Tenim el següent vocabulari: {taula, gos, paper, aigua}
La mida del vector resultant d'aquest vocabulari és de 4 i podem assignar les paraules a la nostra codificació de la següent manera:
taula -> 1000
gos -> 0100
paper -> 0010
aigua -> 0001

Aquest mètode és especialment útil a l'hora de treballar amb models de *natural language processing,* ja que ens permet de manera instantània crear una codificació per un vocabulari en concret o trobar la codificació associada a una paraula. Fins i tot es pot crear "sobre la marxa" simplement assignant una codificació creixent a mesura que anem trobant paraules noves i anar emmagatzemant-les a un diccionari.

6.2.2. Bag of Words

Consisteix a enumerar la quantitat de paraules dins d'una sentència, tenint clau - valor amb clau = la paraula , valor = el nombre d'ocurrències. És molt simple de representar amb un *hashmap* o amb un format *json*. No conserva la gramàtica ni la composició de les sentències[21].

[21] "Bag-of-words model - Wikipedia." https://en.wikipedia.org/wiki/Bag-of-words_model. Es va consultar el 14 abr. 2022.

6.3. Tipus de models

El model és la tècnica que fem servir per fer les associacions de paraules, normalment els models estan encapsulats en llibreries que fan que sigui més fàcil manegar-los i fer operacions amb aquests.

En aquest projecte s'han fet servir dos tipus de models.

6.3.1. Word2Vec

És un concepte per generar representacions vectorials a les paraules. Es dóna una interpretació numèrica per paraula per tal de poder deduir relacions entre paraules dins de tot el model. En *word2vec* aquesta interpretació no es dóna només per una simple codificació, sinó que es recorre a representacions multivectorials.

6.3.2. Doc2Vec

És similar a *word2vec*, però fent servir un vector de document, com a *word2vec*, hi ha dos tipus d'algorismes per donar forma a aquestes representacions.

6.4. Reducció de dimensionalitat

Existeixen algorismes de reducció de dimensionalitat que el que fan és prendre una funció de n dimensions a x dimensions on n > x, i on qualssevol dos objectes en n dimensions i amb una distància propera, a l'aplicar la reducció de dimensionalitat la distància continuarà sent semblant. És molt útil quan treballem amb vectors de dimensions molt grans i volem estudiar com es comporten aquests vectors dins l'espai, fent accessibles visualitzacions en dos i tres dimensions.

6.4.1. t-SNE

Algorisme no lineal i adaptable a informació subjacent, fa diferents transformacions en diferents regions, és estocàstic, fent que l'execució consecutiva d'aquest algorisme dóna diferents resultats. Té un hiperparàmetre anomenat perplexitat i un altre que es diu epsilon, aquest segon indica el *learning rate*[22].

La perplexitat indica com es defineix el balanç entre els aspectes local i globals de les dades, per exemple, si tenim punts en agrupacions molt aïllades entre si, si fem servir una perplexitat molt baixa el resultat en dues dimensions serà que les agrupacions seran de molt pocs punts i molt barrejades

[22] "How to Use t-SNE Effectively - Distill.pub." 13 oct. 2016, https://distill.pub/2016/misread-tsne/. Es va consultar el 14 abr. 2022.

perquè se li donarà més importància al més proper que tingui, en canvi, si el configurem a una perplexitat més alta llavors tindrem agrupacions molt més grans i separades[23].

6.4.2. UMAP

L'algorisme construeix un graf amb pesos de dimensionalitat alta, les arestes representen la versemblança entre dos punts que siguin connectats, per determinar la connectivitat el que es fa és anar estenent un radi per cada punt, els radis que col·lapsin es connecten. Una vegada fet això es construeix un graf de dimensionalitat baixa a partir del generat fent que siguin tan semblants com sigui possible, i d'aquí ja es poden visualitzar les dades[24].

Es fan servir dos hiperparàmetres anomenats nombre de veïns *n_neighbors* i mínima distància entre *low-dimensional points*.

N_neighbors és el nombre aproximat de veïns propers que es fan servir per construir el graf d'alta dimensionalitat, valors baixos fan que l'algorisme prioritzi més les estructures més locals, fent que s'agrupin veïns propers només, valors alts prioritzen estructures més globals, fent que veïns molt llunyans s'agrupin.

Min_dist controla els punts agrupats, valors baixos fan que els punts de cada grup estiguin molt més propers entre sí, valors alts agrupen punts amb més pèrdues fent que es preservi millor l'estructura topològica.

UMAP és molt més ràpid que *t-SNE*.

6.5. Classificació

Els algoritmes de classificació ens ajuden a tenir una representació de rellevància d'un conjunt de documents, en aquest treball presentarem només un d'ells.

6.5.1. tfidf

És una classificació de paraules per documents per tal de veure quines paraules són les més rellevants dins d'un document a tot un corpus. S'ha programat un algorisme mitjanament eficient per tal de poder fer les proves, però no és escalable, ja que la màquina des de la qual es fan les proves no dóna per molt més.

L'algorisme rep un *corpus*, un tipus: *List<List<(Any, List<String>)>>*, això representa un *corpus*, un llistat de llistats de documents, es va decidir afegir un nivell de llista per poder fer més endavant una

[23] "t-distributed stochastic neighbor embedding - Wikipedia." https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding. Es va consultar el 14 abr. 2022.
[24] "Understanding UMAP." https://pair-code.github.io/understanding-umap/. Es va consultar el 14 abr. 2022.

diferenciació per classe de document (per exemple, grups d'una temàtica i grups d'una altra, escalable per n temàtiques). El tipus *Any* representa qualsevol cosa que es vulgui afegir des de qui crida la funció (per si es vol conservar informació addicional, per exemple, el nom del document). Hi ha dos paràmetres més, *top_words* i *num_docs*, el *top_words* és el nombre de paraules que agafarà definitivament per cada document classificat per ordre descendent de rellevància al *corpus*. El *num_docs* el que fa és fitar el nombre de documents de cada temàtica per tal de no agafar tot el *corpus* a posteriori.

L'algorisme s'ha optimitzat, calcula primer tots els *idfs* i ho emmagatzema a una *hashtable*. Per cada document, calcula les ocurrències de totes les paraules al document i ho emmagatzema a una *hashtable* perquè posteriorment, els accessos siguin amb cost constant.
Busca el terme amb més ocurrències que ho fa servir per calcular el tf.
Amb tot això, el bucle interior té un cost constant per iteració.

6.6. Contexts

El context és el que fem servir com a model sobre el qual calcular els *embeddings*, un model és calculat d'acord amb un conjunt de textos, aquests textos poden tindre una temàtica o no, en aquest projecte hem entrenat 3 models amb textos de diferents categories.

Cada context diferent representa un espai vectorial diferent, on cada paraula estarà en un lloc diferent de cadascun d'aquests espais, per tant, calcular el mateix *document embedding* al context científic i calcular un *embedding* dins del context de blogs ens donarà mesures bastant diferents. És important recalcar la importància d'això perquè en els assaigs que es faran es tindrà molt en compte aquest factor.

Cada dataset ha sigut preprocessat abans amb la *pipeline* comentada en anteriors seccions.

6.6.1. Científic

Aquest context té la particularitat de què conté termes molt menys comuns al llenguatge natural, els documents són una mica més llargs, el vocabulari és molt més científic, no es fan servir termes com els d'una conversació comuna, això fa que si per exemple fem servir conversacions normals per aquest context la precisió serà bastant baixa.

6.6.2. Blogs

És una recopilació de conversacions d'antics fòrums, es fa servir un llenguatge més comú. Com per exemple es fa referència a objectes, emocions, noms comuns i noms propis, ubicacions, etc.

6.6.3. Wikipedia

És el context més general, amb aquest podrem cobrir un gran percentatge de documents que es comportin bé amb aquest context. Evidentment, si volem precisió necessitarem contextos més específics.

6.7. Mesures bàsiques utilitzades

En aquest projecte una de les coses més fonamentals és la mesura de distàncies entre *embeddings*, per mesurar una distància es poden fer servir diversos mètodes, depèn de la representació de dades que tinguem, en el nostre cas estem sempre a l'espai Euclidià, això implica que algunes mètriques són impossibles d'aplicar sense transformacions que afectin el resultat final com per exemple la mètrica de Jaccard i el Sorensen Dice.

La mètrica de Hamming teòricament tampoc es podria fer servir, però l'hem adaptat per veure com es comporta, que ja avancem que no es comporta massa bé.

6.7.1. Euclídea

La mètrica euclidiana[25] entre dos punts a un espai euclidià, és la longitud d'un segment de línia entre els dos punts. En el cas on tenim vectors hem de recórrer a la norma euclidiana on s'admet un espai vectorial. En el cas de voler mesurar la distància de dos vectors, cada component dimensional dels dos vectors els haurem de restar en comptes d'agafar el component complet.

Calculat amb la llibreria de *numpy*, `np.linalg.norm(a - b)`

El temps de computació és de **O(m)** on m = el nombre de dimensions dels vectors.

6.7.2. Cosine similarity

És una mesura de similaritat[26] entre dos vectors no nuls en un *inner product space*. Aquest espai es defineix com a un espai real de vectors o un espai complex de vectors amb una operació binària anomenada *inner product*. El *inner product*[27] de dos vectors a l'espai és un escalar.

Llavors el *cosine similarity* és la operació de cosinus de l'angle entre dos vectors, aquest és també el mateix al *inner product* dels mateixos vectors normalitzat.
Calculat amb la llibreria *scipy*, *spatial.distance.cosine*
Es donen dos vectors i ho calcula.
El cost temporal per calcular és:

O(innerP), on n = el nombre de vectors
O(innerP) = O(2m + 2norm(m)) , on m és el nombre de dimensions
O(norm(m)) = O(2m) , és la norma euclidiana
Per tant
Cost = O(2m + 2*2m) = O(2m + 4m) = **O(6m)**

[25] "Euclidean distance - Wikipedia." https://en.wikipedia.org/wiki/Euclidean_distance. Es va consultar el 14 abr. 2022.
[26] "Cosine similarity - Wikipedia." https://en.wikipedia.org/wiki/Cosine_similarity. Es va consultar el 14 abr. 2022.
[27] "Inner product space - Wikipedia." https://en.wikipedia.org/wiki/Inner_product_space. Es va consultar el 14 abr. 2022.

6.7.3. Word Mover Distance

Mètrica basada en el problema del *Earth Mover Distance* i amb l'algoritme de *Max Flow*. Observació important, només es calcula distàncies entre documents, no entre paraules, de fet, aquesta mètrica fa servir altres més simples per computar una distància global. Consisteix en diverses fases (l'explicació està basada en el *paper* del Kusner[28]):

6.7.3.1. Word2Vec Embedding

S'assumeix que les sentències a calcular la distància amb aquesta mètrica són *embeddings*, concretament *word2vec embeddings*, a la pràctica es pot fer servir qualsevol representació en *embedding*, amb això tenim un vector de pesos que farem servir més endavant.

6.7.3.2. Representació en normalized Bag of Words (nBOW)

La idea és poder fer servir els pesos proporcionats pels embeddings donats de *word2vec* i passar a representació de *nBOW* abstraient el detall que cada paraula és un vector, això es pot fer emmagatzemant dos valors a cada paraula, un el *word count* i l'altre el vector que representa la paraula, òbviament amb clau la paraula.

6.7.3.3. Word travel cost

Volem associar la similaritat entre parells de paraules, podem fer servir qualsevol mètrica amb la qual puguem calcular una distància entre dos vectors. Normalment, es fa servir la mètrica euclidiana.

6.7.3.4. Document Distance

Hem de representar aquestes distàncies de parells de paraules com un graf de *max flow* (representació en matriu), sigui **d** la representació *nBOW* del primer document i **d'** la representació del segon document . Cada posició de la matriu diu quant es necessita de la paraula **x** a **d** per viatjar a la paraula **y** a **d'**, per tant, el sumatori de tots aquests pesos d'una columna (sumar tots els pesos del document **d** per arribar a una paraula **y** a **d'**) ha de ser el *outgoing flow* des de la paraula x igual a la paraula **x** a **d**. També, el *incoming flow* de la paraula **y** (sumatori dels pesos del document **d'** per arribar a una paraula **x** a **d**) ha de coincidir amb la paraula **y** a **d'**. Per tant, definim la distància entre dos documents com el mínim cost acumulatiu requerit per moure totes les paraules des de **d** a **d'**.

6.7.3.5. Referència a Earth Mover Distance (EMD)

El problema i solució està explicat, però la manera més simple de resoldre'l no, i és aplicant tres restriccions com un problema de programació lineal, per tant[29]:

---

[28] "From Word Embeddings To Document Distances - Proceedings of ...." https://proceedings.mlr.press/v37/kusnerb15.pdf. Es va consultar el 14 abr. 2022.
[29] "Word Distance between Word Embeddings | by Edward Ma." 25 ago. 2018, https://towardsdatascience.com/word-distance-between-word-embeddings-cc3e9cf1d632. Es va consultar el 14 abr. 2022.

$$\min_{\mathbf{T} \geq 0} \sum_{i,j=1}^{n} \mathbf{T}_{ij} c(i,j)$$

$$\text{subject to:} \sum_{j=1}^{n} \mathbf{T}_{ij} = d_i \quad \forall i \in \{1, \ldots, n\}$$

$$\sum_{i=1}^{n} \mathbf{T}_{ij} = d'_j \quad \forall j \in \{1, \ldots, n\}.$$

Figura 3: restriccions Word Mover Distance, extret de la publicació de Kusner.

d_i és la paraula **x** a **d**
d_j és la paraula **y** a **d'**
T_{ij} és el valor corresponent a la paraula **x** i **y** de la matriu *max flow*.
c(i, j) és la distància euclidiana calculada dos apartats abans.

Les optimitzacions imposades a dalt és un cas especial del *EMD*[30].

El cost d'aquest algorisme en el cas mitjà és **O(p^3 log(p))** on **p** és el nombre de paraules úniques.

6.7.4. Manhattan

Aquesta mètrica[31] està basada en una forma de geometria on la funció de distància de la geometria euclidiana és reemplaçada per una nova mètrica on la distància entre dos punts és la suma absoluta de les diferències de les seves coordenades cartesianes.

El temps de computació és de **O(m)** on m = el nombre de dimensions dels vectors.

6.7.5. Chebyshev

La mètrica de *Chebyshev*[32] és una mètrica definida a l'espai vectorial on la distància entre els dos vectors és la màxima de les diferències absolutes de qualsevol coordenada de cada dimensió.

El cost és lineal **O(m)** on m són les dimensions.

[30] "[2105.14403] Re-evaluating Word Mover's Distance - arXiv." 30 may. 2021, https://arxiv.org/abs/2105.14403. Es va consultar el 14 abr. 2022.
[31] "Taxicab geometry - Wikipedia." https://en.wikipedia.org/wiki/Taxicab_geometry. Es va consultar el 14 abr. 2022.
[32] "Chebyshev distance - Wikipedia." https://en.wikipedia.org/wiki/Chebyshev_distance. Es va consultar el 14 abr. 2022.

6.7.6. Minkwoski

És una generalització[33] de la mètrica euclidiana basada en un hiperparàmetre p.
Si **p = 0** => mètrica euclidiana
Si **p = inf** or **-inf** => mètrica manhattan
Amb altres valors de p es poden donar altres interpretacions.

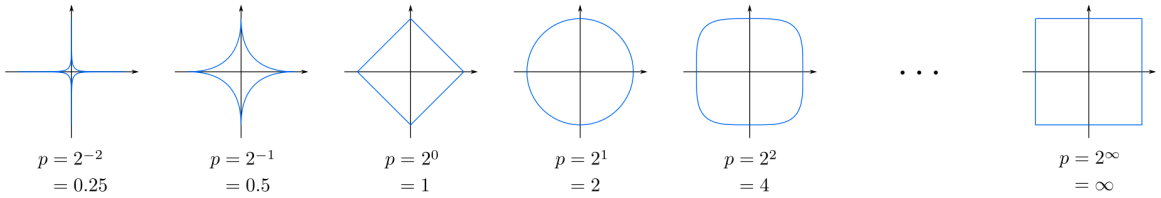

Figura 4: comportament hiperparàmetre p de Minkowski, extret de la wikipedia.

La mètrica de *Minkowski* d'ordre p (on p és un enter) entre dos punts:

$$X = (x_1, x_2, \ldots, x_n) \text{ and } Y = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n$$

Figura 5: assignació de variables prèvia al càlcul de la mètrica Minkowski, extret de la wikipedia

és definida com:

$$D(X, Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}.$$

Figura 6: mètrica Minkowski entre dos punts, extret de la wikipedia.

Al projecte es prova amb diversos valors de p, concretament p in {2, 3, 4, 8, 16}

El cost de computar aquest algorisme és de **O(m)** on m és el nombre de dimensions.

6.7.7. Haversine

Haversine[34] només treballa en dues dimensions, ja que només fa servir latitud i longitud. El que fem és reduir dimensionalitat amb *tsne* i després interpretar **x** com a latitud i **y** com a longitud.

Podem imaginar aquesta mètrica com la distància que recorrem al planeta terra entre dos punts, mai serà una línia recta, serà com si estiguéssim caminant per la superfície d'una esfera (el cas on la latitud és constant).

[33] "Minkowski distance - Wikipedia." https://en.wikipedia.org/wiki/Minkowski_distance. Es va consultar el 14 abr. 2022.
[34] "Haversine formula - Wikipedia." https://en.wikipedia.org/wiki/Haversine_formula. Es va consultar el 14 abr. 2022.

L’explicació del càlcul d’aquesta mètrica queda fora del context del projecte, es poden consultar les fonts citades, proporcionen una explicació més en profunditat.

El cost d’aquest càlcul és constant **O(1)**[35]**,** ja que només es pot calcular en dues dimensions, en cas que el cost de computació de sinus, cosinus i arcsinus fos diferent del cost constant (a la màquina es fan aproximacions per sèries de *taylor*, per tant, podríem agafar l'agregació dels tres costos) podem menystenir-ho.

[35] "How can I quickly estimate the distance between two (latitude ...." 1 abr. 2013, https://stackoverflow.com/questions/15736995/how-can-i-quickly-estimate-the-distance-between-two-latitude-longitude-points. Es va consultar el 14 abr. 2022.

6.7.8. Hamming

Hamming[36] és una mètrica basada en comparacions entre tires de 0's i 1's, mesura les discrepàncies posicionals i ens dóna una xifra que ens indica la diferència entre els dos números, convé sempre normalitzar d'alguna manera, o bé afegint tants zeros davant com la xifra més llarga o bé donant un valor entre un rang concret (per exemple entre 0 i 1).

Per aplicar-la al nostre projecte el que s'ha fet ha sigut:

Conversió a binari de totes les dimensions de cada vector (cada element de cada dimensió)
Per exemple, tenim **x = [x1, x2, x3, … xn] => binary([x1, …]) => [binary(x1), …]**

Després fer **hamming(x1, y1) + hamming(x2, y2) + ...**
Amb això tenim la distància per cada dimensió i la sumem, mètode molt semblant a distància euclidiana, es fa la suma de quadrats.

Característiques de la conversió a binari: *big-endian* (irrellevant pel cas), en format *double* (ens ho podem permetre).

Ho hem fet així pel fet que *hamming distance* només té sentit amb cadenes de caràcters d'igual llargària, es pot arribar a fer amb enters, sempre que es comenci per la dreta cap a l'esquerra, també tenir en compte la base del sistema de numeració, si fem sistema binari farem que se sumin més distància entre caràcters, si fem *base 64* la distància serà molt més petita entre paraules molt diferents, ja que *hamming* és una mesura posicional.

No s'han fet proves amb bases numèriques diferents de 2, queda fora de l'abast del projecte.

El cost de fer la distància de *Hamming* és el cost de passar cada coordenada del vector a binari multiplicat pel nombre de bits que representa cada número, ja que hem de comptar la diferència posicional de cada bit de cada coordenada amb el vector que volem comparar.

Per expressar el cost de passar a binari hem de desglossar el procés de passar a binari un nombre, té diverses fases:

- Passar a *bytes* el *float*, a *python* els *floats* ocupen 8 *bytes*, això no té cost, ja que només estem fent una conversió que a memòria ja hi és. => **O(sizeof(float))**
- Aplicar el format *string* als *bytes* donats, això és lineal respecte el nombre de bits, ja que només cal recórrer bit a bit i emplenar una string. => **O(n) n = nb de bits**
- Reemplaçar el '0b' per res, lineal respecte el nombre de caràcters. => **O(c) c = *string.length***
- Justificar els zeros de davant de cada *byte*, lineal respecte el nombre de *bytes*. => **O(c) c = *string.length***

Una vegada tenim cada coordenada del vector a binari, hem de comparar coordenada a coordenada, bit a bit quins difereixen, en aquest cas cada bit és equivalent a un caràcter. Això té un cost de **O(2nc)** on **n** és el nombre de dimensions i c el nombre de bits de cada dimensió.

[36] "Hamming distance - Wikipedia." https://en.wikipedia.org/wiki/Hamming_distance. Es va consultar el 14 abr. 2022.

Recapitulant, el cost de tot això és:

O(sizeof(float)) + O(n) + O(c) + O(c) + O(2nc) =>
O(sizeof(float) + n + 2c + 2nc) =>
O(2nc) // 2nc > n > 2c > sizeof(float)

7. Preprocessament de text i anàlisi

Una vegada tenim una imatge global d'aquesta branca de coneixement, com hem vist a l'estat de l'art, i hem repassat els conceptes més bàsics que es fan servir en aquest projecte, ja podem procedir a veure com s'ha desenvolupat tot el projecte.

Començarem pel preprocessament de text i l'anàlisi dels datasets, es comença primer pel preprocessament de text perquè és més senzill fer una anàlisi d'un dataset ja preprocessat perquè les paraules ja són a la seva forma canònica.

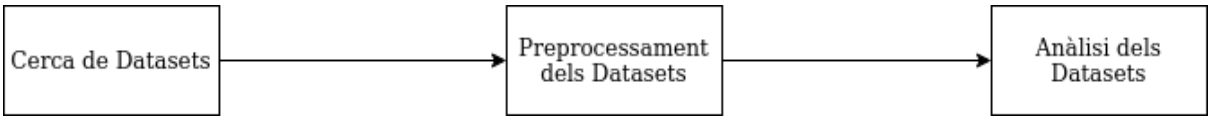

Figura 7: cicle del preprocessament de text i anàlisi.

En aquesta secció s'expliquen els procediments que s'han aplicat a l'hora d'estructurar el projecte i els diferents passos.

Cal afegir també que el preprocessat de documents llegeix d'una ubicació, preprocessa el document i guarda el resultat a una altra carpeta per tal de poder-los fer servir més tard per entrenar els models o fer-los servir per als experiments.

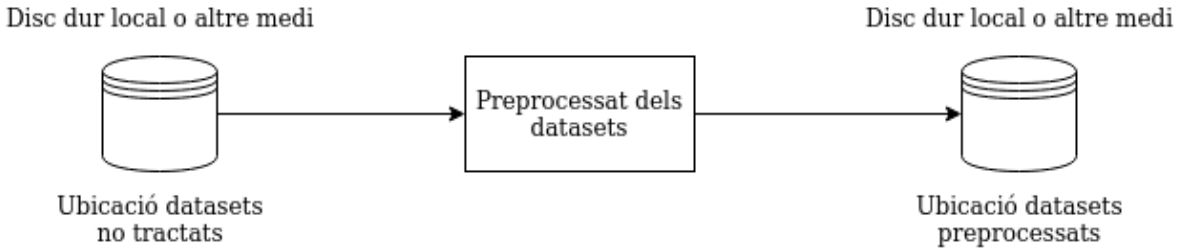

Figura 8: lectura datasets, processament i escriptura.

7.1. Obtenció de datasets

En aquesta secció s'expliquen els datasets que s'han fet servir al projecte i una petita anàlisi de cadascun.

7.1.1. Dataset documents científics

Agafat d'un dataset de pdfs de papers científics i convertits a text pla, és el context científic, els *papers* són sobretot de tecnologies de la upc. La mida d'aquest conjunt de documents és de 1180 documents.

7.1.2. Dataset entrades de blogs

El dataset corresponent a aquest context[37] es va agafar de la xarxa, és un recopilatori d'entrades de blogs formatades a text. Aquest context és més "casual" i marca bastant diferència amb el científic. La mida d'aquest dataset són 18000 entrades de blogs, el que la mida de cada entrada és molt més reduïda que un document científic en comparació.

7.1.3. Dataset wikipedia

Aquest és el context més general, és amb Word2Vec[38] i Doc2Vec[39], ja que wikipedia conté entrades de totes les temàtiques existents. Els datasets existents eren massa gran ocupant molts GB d'espai, una mida no assumible amb els recursos informàtics dels quals disposem, ja que després s'ha d'entrenar el model i no és assumible tindre l'ordinador treballant sense parar dues setmanes. Per tant, es va agafar un model ja entrenat.

7.2. Preprocessament de text

Abans de fer servir qualsevol document hem de preprocessar-lo correctament per tal que els models funcionin correctament.

A *NLP* quan parlem de paraules hem de "col·lapsar" aquelles que semànticament vulguin dir el mateix, ja que a l'hora de fer servir aquestes, és possible que perdem precisió. Per exemple, si tenim moltes conjugacions d'un verb a un mateix document, cada vegada que aparegui aquesta paraula que semànticament diu el mateix, es relacionarà amb les que tingui al voltant i el model la "posicionarà" de manera que estigui bastant separada de les altres conjugacions que en realitat volen dir al mateix, separada en el pla geomètric que representa un embedding. Per tant, hi han maneres de fer que un verb conjugat de moltes maneres ho transformem a una sola paraula **canònica**, s'explicarà més endavant.

A més, també cal treure paraules que no vulguin dir absolutament res en si ni tan sols si la fas servir juntament amb altres paraules, com per exemple, els articles. Treure aquesta mena de paraules ens ajuda a apropar el model als conceptes més importants, en cas de deixar aquesta mena de paraules que normalment es repeteixen bastant farà que tinguin més pes al model deixant de banda altres conceptes que podrien ser molt més descriptius.

Amb tot això cal destacar que també hem de treure signes de puntuació, urls, etiquetes, etc. Tampoc ens diuen res a l'hora de fer una anàlisi semàntica.

---

[37] "Blog Authorship Corpus - Cohen Courses." 2 nov. 2011, http://curtis.ml.cmu.edu/w/courses/index.php/Blog_Authorship_Corpus. Es va consultar el 14 abr. 2022.
[38] "Pretrained Embeddings - Wikipedia2Vec." https://wikipedia2vec.github.io/wikipedia2vec/pretrained/. Es va consultar el 14 abr. 2022.
[39] "jhlau/doc2vec: Python scripts for training/testing paragraph vectors." https://github.com/jhlau/doc2vec. Es va consultar el 14 abr. 2022.

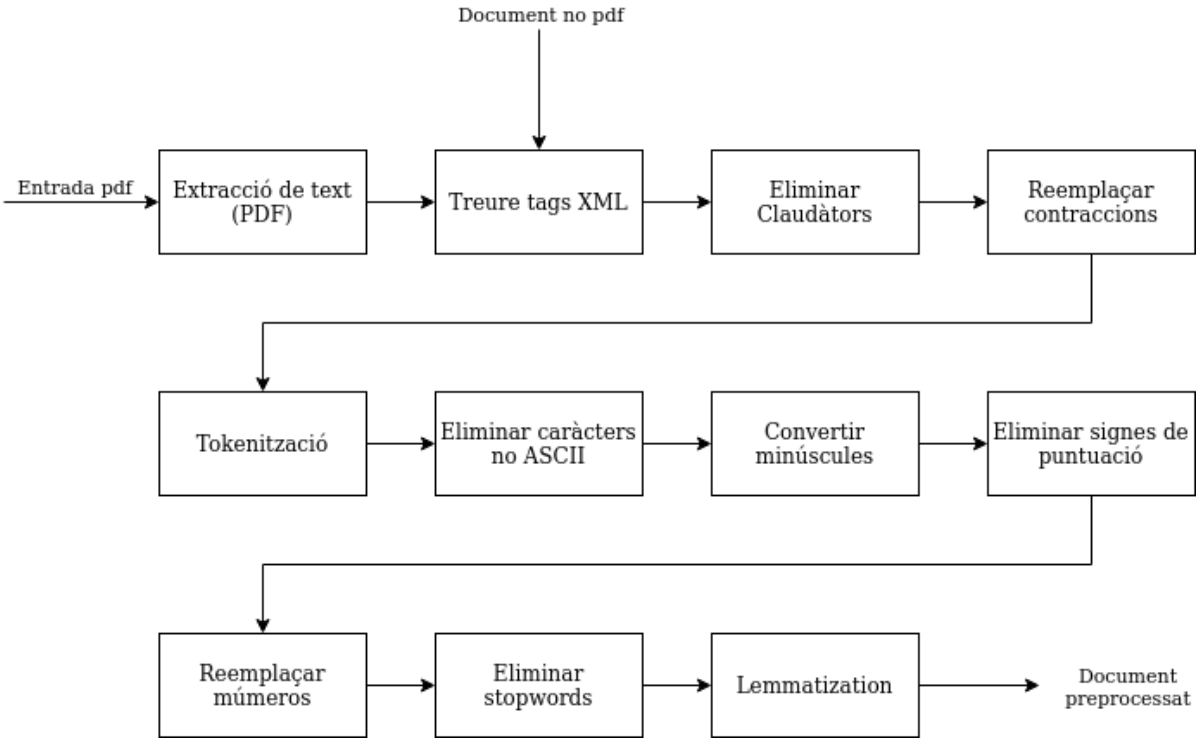

Figura 9: pipeline de preprocessament de text.

7.2.1. Extracció de text

Tenim tres tipus de documents, *pdfs* dels documents científics, *xml* i text dels documents de blogs, l'únic tipus que requereix un pas previ són els arxius *pdfs*.

7.2.1.1. PDF

És molt comú avui dia haver d'extreure text de documents *pdfs* això dóna peu al fet que existeixin eines molt fàcils de fer servir per fer aquesta finalitat en comptes del que s'ha fet sempre que era veure com és l'estructura d'un fitxer *pdf* i operar amb bytes amb molta cura per anar transformant a text.

L'eïna que hem fet servir al projecte és *tika*[40], és de molt fàcil ús, té una funció *from_file* on l'argument és el *path* d'un fitxer, en aquest cas *pdf*, el retorn d'aquesta funció és un diccionari amb totes les dades i metadades del fitxer, aquest té una clau *'content'* que ens donarà el text pla d'aquest fitxer *pdf*.

[40] "Parsing PDFs in Python with Tika - Clinton Brownley's Decision ...." 26 jun. 2016, https://cbrownley.wordpress.com/2016/06/26/parsing-pdfs-in-python-with-tika/. Es va consultar el 14 abr. 2022.

7.2.2. Pipeline de preprocessament de text

Hem d'aconseguir reduir al màxim de possibles variants les paraules que hi hagin al text, també hem de trobar una manera de representar números i signes *unicode* a *ascii*; amb l'objectiu de poder crear una codificació única per paraula i la més canònica possible.

Tot el text que introduïm als models a entrenar han d'anar separats per espais, si per exemple deixem un punt en aquesta paraula a l'hora de crear una codificació pel vocabulari aquesta comptaria com una nova paraula, per tant, al vocabulari constaria la paraula sense punt i la paraula amb punt, i molt possiblement en ser una paraula que només apareix una vegada a l'hora de crear la gràfica del model en dues dimensions o bé calcular la distància entre aquestes dues paraules les veuríem probablement més separades entre si que d'entre altres paraules amb l'original. Això és degut al fet que aquesta paraula amb punt apareix una vegada o més (depèn el text això), però aquesta freqüència probablement és d'un ordre de magnitud inferior a l'original, això fa que a l'hora d'entrenar el model les paraules que va trobant al seu voltant (significat semàntic que se li dóna basant-se en les paraules del seu voltant) difereixi probablement bastant de l'original, perquè a més, en aquest cas, si és punt sempre hi serà al final d'una frase.

7.2.2.1. Treure tags xml

El primer pas de tots és treure els tags d'xml que hi hagi al text. Aquests *tags* de cara a la informació que volem donar-li al model no ens diu absolutament res.

Ho podem fer amb una eina molt simple anomenada *BeautifulSoup*[41], és una llibreria per tractar aquest tipus de format, a més de fer moltes altres coses ens permet obtenir el text pla d'una string en format *xml*.

7.2.2.2. Eliminar claudàtors

Els claudàtors solen indicar informació que no és rellevant com números i cites, de cara a la informació que volem extreure del text no és gens rellevant i a més esbiaixa el context de la frase fent que pugui significar una cosa totalment diferent. Per això és important fer una neteja de tots aquests claudàtors.

Per fer aquesta neteja ens podem ajudar d'una expressió regular que faci *match* de tot el que hi hagi entre claudàtors, en el cas de *python* hem fet servir la següent expressió regular la qual substitueix per una string buida els claudàtors i el que hi hagi dins: '[[^]]*]'

[41] "Beautiful Soup 4.9.0 documentation - Crummy." https://www.crummy.com/software/BeautifulSoup/bs4/doc/. Es va consultar el 14 abr. 2022.

7.2.2.3. Reemplaçar contraccions

Aquest és un pas específic de l'Anglès, ja que aquest idioma té contraccions d'una o més paraules que fan que un conjunt de paraules enllaçades segons amb quin tipus de signes de puntuació es puguin escriure amb una única paraula o un grup de paraules més canòniques.

Per exemple, si tenim *"i've"* el podem substituir per *"i have"*, seran dues paraules diferents, però que a causa de la seva proximitat els models a l'hora de fer l'entrenament la tindran en compte i, per tant, semànticament conservarà el seu significat en conjunt.

Cal destacar que cada idioma té les seves particularitats respecte a això i potser s'han d'implementar passos addicionals, per exemple al català tenim també contraccions i si volem conservar el significat d'una paraula amb dependència de les seves regles d'escriptura com els accents o les eles geminades haurem d'aplicar més passos.

Per fer aquesta operació hem fet servir una llibreria anomenada *contractions*[42], simplement li passem un text i ja ens dóna el retorn corregit.

7.2.2.4. Tokenització

És el procés de segmentar en unitats un text, les unitats que fem servir són les paraules i és la granularitat més petita que podem fer servir, ja que una lletra o un conjunt de lletres dins d'una paraula no ens aporten absolutament cap informació.

No és tan trivial com pot semblar, ja que ara mateix tenim al text tots els signes de puntuació que normalment es troben tot just darrere o davant d'una paraula, cometes, parèntesis, guions, etc. Per tant, no es pot fer simplement separant per espais.

La tokenització és el pas més fonamental de qualsevol processament de text, el que es fa és anar emmagatzemant els diferents tokens en un Arbre Abstracte de Sintaxi (AST) per tal de poder capturar els blocs lògics i més tard poder accedir en aquest arbre per tal de fer altres operacions, sobretot és molt fet servir a l'hora de fer l'anàlisi sintàctica d'un llenguatge de programació qualsevol, en el nostre cas serà molt més simple i l'alçada i ramificació d'aquest serà donat pels diferents parèntesis, cometes, signes, etc.

Existeix una llibreria que ens facilita això i no haver d'escriure totes les expressions regulars i contemplar absolutament tots els casos, parlem de *nltk - tokenize*[43].

Exemple de funcionament[44]:

'"Good muffins cost $3.88\nin New York. Please buy me\ntwo of them.\nThanks."'

---

[42] "contractions - PyPI." https://pypi.org/project/contractions/. Es va consultar el 14 abr. 2022.
[43] *"nltk.tokenize package." https://www.nltk.org/api/nltk.tokenize.html. Es va consultar el 14 abr. 2022.*
[44] "nltk/__init__.py at develop · nltk/nltk - tokenize - GitHub." https://github.com/nltk/nltk/blob/develop/nltk/tokenize/__init__.py. Es va consultar el 20 abr. 2022.

['Good', 'muffins', 'cost', '$', '3.88', 'in', 'New', 'York.', 'Please', 'buy', 'me', 'two', 'of', 'them.', 'Thanks', '.']

"They'll save and invest more."
['They', "'ll", 'save', 'and', 'invest', 'more', '.']

"hi, my name can't hello,"
['hi', ',', 'my', 'name', 'ca', "n't", 'hello', ',']

7.2.2.5. Eliminar caràcters no ASCII

Els models que fem servir només funcionen amb caràcters de codificació *ascii*, això vol dir que qualsevol altre caràcter que tingui una codificació diferent no serà interpretat com realment és, per exemple, si fem servir un caràcter *unicode*, dins de l'espai de codificació aquest ocupa 1 bytes fins que s'excedeix la mida possible de 1 *bytes*, llavors passa a 2 *bytes*, quan se supera aquesta mida passa a 3 bytes, etc. *Ascii* només funciona amb 255 possibles caràcters, per tant, si intentem forçar *unicode* als models el que passaria és que només interpretaria *byte* a *byte* i evidentment no serien els que representen el caràcter que hi ha realment[45].

Hem de diferenciar entre dos conceptes, una és la manera de formar un caràcter (*ascii* o *unicode*) i altre és la manera de representar-ho a un fitxer (*utf-8*). *Utf-8* és compatible amb *unicode* i *ascii* pel fet que fa servir escriptura de caràcters de mida variable, i els primers 255 caràcters d'*unicode* són els mateixos que *ascii*.

Aquest pas té més d'una solució, aquí proposem dos, però en fem servir només una.

La primera és deixar-ho com està, si una paraula és composta per caràcters *unicode* per exemple, representant una paraula en xinès, podem deixar-ho com està pel fet que a l'hora de crear el vocabulari pel model s'agafarà com una nova paraula la combinació de *bytes* de cada paraula en xinès que fem servir, com que a *unicode* la representació és única per caràcter, la formació de grups de caràcters també serà única i, per tant, cada combinació diferent comptarà com una nova entrada al vocabulari del model. El desavantatge principal d'aquest mètode és que s'haurà de fitar els caràcters *unicode* que realment ens representi la semàntica del que volem entrenar, per exemple, els emojis no ens diuen res (a priori, en cas que només volguéssim contemplar un alfabet concret, però sí que es podria arribar a fer servir) o inclús es podria canviar cada *emoji* amb una paraula que representés *l'emoji* (convertint-lo a forma canònica).

La segona és eliminar els caràcters que sobrepassin el 255, si estem treballant amb l'anglès no té sentit contemplar caràcters xinesos o japonesos o bé crear una traducció per tots aquests (però per fer això s'hauria de fer un pas previ a la tokenització). El gran desavantatge d'aquest mètode és que es dóna una pèrdua d'informació eliminant aquests caràcters, ja que si hi ha un *emoji* és que la persona volia donar informació addicional a la frase o bé en sí volia dir alguna cosa concreta amb aquest. La solució que farem servir al projecte és aquesta, ja que tot el text que fem servir és anglès.

[45] "What is the advantage of choosing ASCII encoding over UTF-8?." 30 jul. 2011, https://softwareengineering.stackexchange.com/questions/97247/what-is-the-advantage-of-choosing-ascii-encoding-over-utf-8. Es va consultar el 14 abr. 2022.

7.2.2.6. Convertir a minúscules

Un dels passos més fàcils d'aplicar, però també dels més importants, un caràcter en minúscula té un codi ascii diferent d'un en majúscula, a l'hora de generar el vocabulari es donarà com una paraula diferent "Teclat" que "teclat", ja que si transformem a *bytes* les dues cadenes de caràcters, el primer codi de les dues cadenes és diferent i essencialment volen dir exactament el mateix.

És possible que en determinats casos vulguem conservar aquesta distinció, depèn molt del context que vulguem crear, a vegades volem donar-li un significat diferent si tenen la primera lletra (per exemple) en majúscula, però en general és molt més simple passar-ho tot a minúscules.

7.2.2.7. Eliminar signes de puntuació

Després del pas de tokenització, els signes de puntuació estan a una sola cadena de caràcters i aïllats de la informació rellevant del text. Per tant ens és molt simple veure quins són i eliminar-los. A efectes de la semàntica del text no ens diuen res, com a molt els signes d'interrogació i exclamació ens podrien arribar a ajudar a entendre el sentiment de la frase, però en general no ens interessa tenir al vocabulari un signe d'exclamació, d'interrogació, etc ja que ens esbiaixa una mica la resta de paraules (pel fet que són sovint bastant freqüents).

Detectar-los es pot fer amb qualsevol expressió regular que detecti qualsevol d'ells.

7.2.2.8. Reemplaçar números

En aquesta fase el que fem és agafar els símbols numèrics [0-9]+ ; i reemplaçar-los per la seva representació textual en anglès. A vegades als textos es fan servir números o lletres per representar els números, tindre a diversos llocs un 5 i un "*five*" no segueix la regla de la forma canònica d'una paraula, a efectes del vocabulari creat pel model són dues paraules diferents.

Per què no agafem la forma en lletres i les passem a números? Perquè és molt més complex determinar si una paraula o conjunt de paraules és un número que un número en si, ja que l'expressió regular per saber si un número, és un número és molt simple (tret de veure si hi ha comes, punts, etc).

Així doncs, existeixen llibreries per fer aquest pas, la que hem fet servir és la de *inflect*[46]. Serveix d'entre altres coses per convertir números a la seva representació textual.

És important fer aquest pas abans de la tokenització pel fet que aquest procés de la tokenització convertiria aquestes representacions en tokens traient-li el sentit a la representació que hem generat.

[46] "inflect · PyPI." https://pypi.org/project/inflect/. Es va consultar el 14 abr. 2022.

7.2.2.9. Eliminar “stopwords”

Les stopwords a *NLP* són aquelles paraules més comunes al llenguatge que encara que les trèiem d’una frase aquesta frase en conjunt continua tenint un significat molt proper.

S’ha de valorar a l’hora de treure cada stopword com és de rellevant per la frase, si per exemple tenim “*This movie is not good*” i li traiem el “*not*” la frase canvia totalment el seu significat, tampoc vol dir que cada vegada que aparegui “not” implica que tinguin un gran pes a la frase. Això és sobretot important a l’anàlisi de sentiments.

Segons la utilitat que li vulguem donar al model ens interessa treure o deixar algunes. En el cas d’aquest projecte no és un apartat molt important, ja que la finalitat dels models que generarem és mesurar posteriorment distàncies entre embeddings i aquest punt de refinament no ens aporta molt.

Per fer aquesta tasca hem fet servir la lliberia *NLTK*[47]. Existeix un mòdul que inclou totes les stopwords possibles en l’idioma anglès, arran d'aquest podem filtrar relativament fàcil les stopwords sense tindre en compte el que hem comentat abans de si el *stopword* està donant un significat concret a la frase, en general no sol fer-ho i per l’abast del projecte no ens cal[48].

Un dels avantatges principals que té treure les stopwords és que la mida del text es redueix significativament, això implica que el temps d’entrenament dels models sigui molt més baix. Un altre és que ens elimina soroll del text fent que el que li donem al model per entrenar-lo sigui més concís i acabi funcionant millor.

[47] "nltk.corpus package." https://www.nltk.org/api/nltk.corpus.html. Es va consultar el 14 abr. 2022.
[48] "Stop Words in NLP - Medium." https://medium.com/@saitejaponugoti/stop-words-in-nlp-5b248dadad47. Es va consultar el 14 abr. 2022.

7.2.2.10. Stemming vs Lemmatization[49]

L'únic que fins ara hem deixat a l'aire és les diferents formes de cada paraula i les famílies de derivacions de cada paraula relacionada.

Quan fem servir les diferents formes d'una paraula en termes del vocabulari del model són paraules totalment diferents pel fet que tenen una codificació diferent. Ens interessa que cada família de paraules les reduïm a una sola forma escrita exactament igual, això implicarà una codificació igual[50].

Existeixen dues maneres de reduir aquestes paraules, una és mitjançant un algorisme general i l'altre és trobar la forma base de la paraula[51].

7.2.2.10.1. Stemming

Stemming[52] usualment es refereix a un procés heurístic que talla un sufix de mida variable de la paraula amb l'esperança de trobar la forma més bàsica de la paraula.

L'algorisme més usual per fer aquest procés es diu "L'algorisme de *Porter Stemming*", hi han d'altres que són més sofisticats o bé que la seva finalitat és fer un stemming una mica diferent, això depèn de la finalitat que tingui el text preprocessat, per nosaltres ens és suficient l'algorisme de Porter.

No explicarem l'algorisme de Porter en detall, ja que és bastant llarg i queda fora de l'abast del projecte, però en resum, tracta de classificar dins les paraules dos grans grups, les consonants i les lletres que no són consonants (a,e,i,o ó u, i y tal que sigui precedida per una consonant). Feta aquesta distinció podem formar regles gramaticals, fer substitucions i aplicar-les per fases mitjançant gramàtiques contextuals, per exemple (extret de la web de Stanford Stemming and Lemmatization):

---

[49] "Stemming vs Lemmatization - Towards Data Science." 13 may. 2020, https://towardsdatascience.com/stemming-vs-lemmatization-2daddabcb221. Es va consultar el 14 abr. 2022.
[50] "What is the difference between lemmatization vs stemming?." https://stackoverflow.com/questions/1787110/what-is-the-difference-between-lemmatization-vs-stemming. Es va consultar el 14 abr. 2022.
[51] "Stemming and lemmatization - Stanford NLP Group." https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html. Es va consultar el 14 abr. 2022.
[52] "The Porter stemming algorithm - Snowball." http://snowball.tartarus.org/algorithms/porter/stemmer.html. Es va consultar el 14 abr. 2022.

Rule:	**Example**:
SSES → SS	caresses → caress
IES → I	ponies → poni
SS → SS	caress → caress
S →	cats → cat

També existeixen substitucions segons la mida de la paraula per exemple:

(m > 1) EMENT →

Això converteix “replacement” to “replac” però no “cement” to “c”.

En total existeixen 5 fases de reducció de paraula aplicades seqüencialment, dins de cada fase trobem diferents convencions per seleccionar regles i sempre s’apliquen per ordre de llargària, és a dir, la regla que inclogui més lletres a substituir sempre serà la primera a aplicar-se, això fa que les regles més curtes que estiguin incloses dins de regles més grans no s’apliquin de manera incorrecta.

El creador de l’algorisme va estendre el seu treball creant el llenguatge *snowball* que serveix per crear algorismes de *stemming*.

Mostrem a continuació uns exemples de les diferents variants d’aquesta família d’algorismes (extret de la web de Stanford Stemming and Lemmatization):

***Sample text**: Such an analysis can reveal features that are not easily visible from de variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation*

***Lovins stemmer**: such an analysis can reve featur that ar not eas vis from th vari in th individu gen can lead to a pictur of expres that is mor biolog transpar and acces to interpres*

***Porter stemmer**: such an analysi can reveal featur that ar not easili visibl from the variat in the individu gene and can lead to a pictur of express that is more biolog transpar and access to interpret*

***Paice stemmer**: such an analys can rev feat that are not easy vis from the vary in the individ gen and can lead to a pict of express that is mor biolog transp and access to interpret*

L’algorisme de Porter és només vàlid pel llenguatge anglès[53], ja que les regles que es defineixen són extretes del mateix llenguatge i si les apliquem al català per exemple no tindrem uns resultats coherents. Segons Porter, el llenguatge Snowball permet crear scripts per fer un stemmer de molts altres llenguatges.

[53] "Does PorterStemmer supports languages other than english?." 14 jul. 2019, https://stackoverflow.com/questions/57026011/does-porterstemmer-supports-languages-other-than-english. Es va consultar el 14 abr. 2022.

Un dels grans avantatges del *stemming* davant el *lemmatization* és que és molt més ràpid ja que aplicar regles gramaticals mitjançant gramàtiques no deixa de ser un algorisme recursiu lineal respecte el nombre de fases i llargària de la paraula.

La gran desavantatge d'aquest mètode és que no totes les paraules tendeixen a tindre una derivació reductible mitjançant extracció de sufixos, per exemple, la paraula "*better*" no es transformarà a "good" que és una forma més bàsica de la paraula.

El *stemming* prioritza el "*recall*" sobre la precisió (*recall* és la mètrica que quantifica el nombre de prediccions positives correctes realitzades a partir de totes les prediccions positives que podrien haver-se donat).

7.2.2.10.2. Lemmatization

La Lematització és la utilització del vocabulari complet i l'anàlisi morfològic de les paraules per trobar la base de la paraula o la forma de diccionari, la qual s'anomena "*lemma*".

El "*Lemma*" també anomenada "forma de diccionari", és la forma base d'una paraula, eliminant qualsevol variació, temps o quantitat.

Exemples (extret de la web de Stanford Stemming and Lemmatization):

- Plural a singular: *girls, boys, corpora → girl, boy, corpus*
- Verbs, variants temps/participi: *ate, brought, chatting → to eat, to bring, to chat*
- Eliminació de gènere: *doctress → doctor*

El funcionament intern de les diverses llibreries que existeixen per fer Lematització inclouen un arxiu de vocabulari (al contrari que el *stemming*, que es fa algorísmicament), aquest vocabulari està etiquetat de manera que per cada forma base es defineixen totes les seves variants i a més s'especifica el tipus de la paraula (determinant, adverbi, pronom, etc). Segons la font que es faci servir cada vocabulari defineix els seus tags, però tots segueixen els mateixos estàndards (el llenguatge és el que és i no es poden canviar aquestes regles).

Una vegada tenim aquest vocabulari podem fer un parser que busqui per cada paraula que vulguem parsejar dins el vocabulari i veure quina és la seva forma base.

Aquests diccionaris no inclouen absolutament totes les formes verbals, normalment es fiten totes les possibles paraules i formes en aquests diccionaris, per la qual cosa a la pràctica es requereixen passos extra de parseig, com per exemple eliminar plurals.

Algorísmicament, no té cap misteri, és una cerca en diccionari i s'apliquen regles molt genèriques. Les llibreries que ja existeixen inclouen tots aquests passos que hem esmentat, al nostre projecte l'hem hagut de descarregar a mà, ja que la llibreria requeria especificar-li un vocabulari concret, no s'adjunten els vocabularis a les llibreries perquè tenen moltes actualitzacions al llarg del temps i tindre-ho dins del codi de la llibreria només crearia una dependència amb el vocabulari dins d'aquesta (imaginem-nos que tenim x vocabularis dins de la llibreria i cadascun d'aquests s'actualitza cada x

setmanes). Podem també escollir quin dels vocabularis volem, per si per exemple volem el vocabulari d'un centre lingüístic concret.

En aquest projecte hem fet servir el *lemmatizer nltk* de *spicy*[54]. El procés general d'un lematitzador normalment és[55]:

1- Busquem les excepcions, agafem el lemma des de la llista d'excepcions si hi és.
2- Apliquem les regles.
3- Guardem els tokens que hi són a la llista d'índexs
4- Si no trobem un lemma als passos 1-3 vol dir que està fora del domini que contemplem (*out-of-vocabulary*) i simplement deixem anar la paraula tal com ha arribat.
5- Retornem les formes en lemma

Respecte al pas 3, la llista d'índexs és una llista intermèdia que fem servir per anar emmagatzemant els resultats, tal que, la clau és la forma original i el valor el lemma convertit.

7.2.3. Assumpcions a posteriori de la pipeline de processament

- Hem convertit a minúscules abans d'eliminar les stopwords, per tant, hem perdut noms que comencen amb majúscules i s'escriuen igual que una *stopword*, per exemple "*The Who*" o bé "*Take That*"
- No hem fet un filtre per paraules molt llargues, a vegades se'ns ha colat alguna paraula que era molt llarga i no volia dir res, fruit del *parseig* dels *pdfs*, ja que no sempre és 100% eficaç.

7.2.4. Resultat final

Hem optat per fer el *Lemmatization* perquè des de ja fa un temps les diferents llibreries de *NLP* han deixat d'incloure el *Stemming* pel fet que el *Lemmatization* en general sempre té millors resultats.

Tot el parseig que hem fet ho hem deixat a un equivalent de fitxers de text, per exemple, si un document el tenim a "path/doc1.txt" l'haurem deixat a "results/doc1.txt", conservant el nom del fitxer[56].

[54] "How does spacy lemmatizer works? - Stack Overflow." 5 may. 2017, https://stackoverflow.com/questions/43795249/how-does-spacy-lemmatizer-works. Es va consultar el 14 abr. 2022.
[55] "How to build a Lemmatizer. And why | Analytics Vidhya - Medium." https://medium.com/analytics-vidhya/how-to-build-a-lemmatizer-7aeff7a1208c. Es va consultar el 14 abr. 2022.
[56] "Preprocess Text in Python --- A Cleaner and Faster Approach." 4 jul. 2019, https://www.thinkdatascience.com/post/preprocess-your-text-for-nlp-models-cleaner/. Es va consultar el 14 abr. 2022.

7.3. Anàlisis dels datasets

Una vegada tenim feta la pipeline podem fer una anàlisi sobre les paraules dels dos datasets, fer-lo abans de preprocessar-los pot donar a obtenir resultats molt poc precisos, ja que dues paraules amb diferent temps verbal donarien dues paraules diferents.

Només podem fer l'anàlisi del dataset de blogs i científic, ja que del de wikipedia hem agafat ja un model preentrenat. Hem agafat diverses mètriques per veure com són els dos datasets i assegurar-nos de què compleixen el que volem, que siguin el prou concrets i diferents.

7.3.1. Primeres 40 paraules més usades

Primer, analitzarem les paraules més usades a cadascun dels datasets, el format és paraula i nombre de vegades que apareix:

7.3.1.1. Científic

and -> 340788
hundred -> 277793
one -> 182279
two -> 179959
thousand, -> 101363
thousand -> 91308
three -> 79009
four -> 60102
five -> 58755
nine -> 49775
six -> 45982
use -> 44108
eight -> 39315
seven -> 37687
de -> 32051
model -> 31964
zero -> 30076
ten -> 28736
c -> 25804
n -> 25585
j -> 25239
p -> 23259
e -> 22913
result -> 22657
al -> 22650
et -> 21437
fifteen -> 20540
eleven -> 20530
twelve -> 19681
http -> 19575

figure -> 19329
b -> 18317
show -> 18258
r -> 18174
fourteen -> 18026
data -> 18001
value -> 17825
thirteen -> 16497
sixteen -> 16391
h -> 16261

Com veiem, es fan servir molts números, nombres de variables (com j, i, a, b, …), nombres de constants i paraules molt sonades com “value”, “http”, “result”, etc.

Per exemple, a la primera barra que quasi a tots els casos succeirà, tenim que la freqüència 1 que es repeteix per 105470 paraules, o sigui, hi ha 105470 paraules que es repeteixen una vegada, després, hi ha 32875 paraules que es repeteixen 2 vegades, etc.

7.3.1.2. Blogs

post -> 1104754
date -> 1047376
go -> 705207
get -> 642178
one -> 574267
like -> 471118
nbsp -> 463147
think -> 365879
know -> 363647
would -> 343166
time -> 341548
say -> 330147
two -> 314240
make -> 296415
urllink -> 281003
and -> 270644
hundred -> 265995
want -> 264016
really -> 260485
see -> 238476
well -> 236546
people -> 226177
come -> 224440
good -> 218043
take -> 214401
back -> 198295
work -> 195318

love -> 191032
day -> 189799
much -> 176157
look -> 176013
feel -> 175596
could -> 172777
even -> 169738
today -> 159517
three -> 157959
us -> 153916
tell -> 153832
way -> 146016
last -> 144855

En aquestes freqüències es veu molt del llenguatge comú i paraules relacionades amb xarxes socials com “post”, “date”, “like”, etc. També podem veure paraules habituals com “day”, “look”; “today”, etc.

7.3.2. Histogrames de freqüències

Els histogrames que descrivim aquí fan referència al nombre de vegades que succeeix que una freqüència es dona. A l'eix **y** tenim el nombre de vegades de la freqüència i a l’eix **x** la freqüència.

7.3.2.1. Científic

A la següent figura es mostra l'histograma de freqüències pel dataset científic:

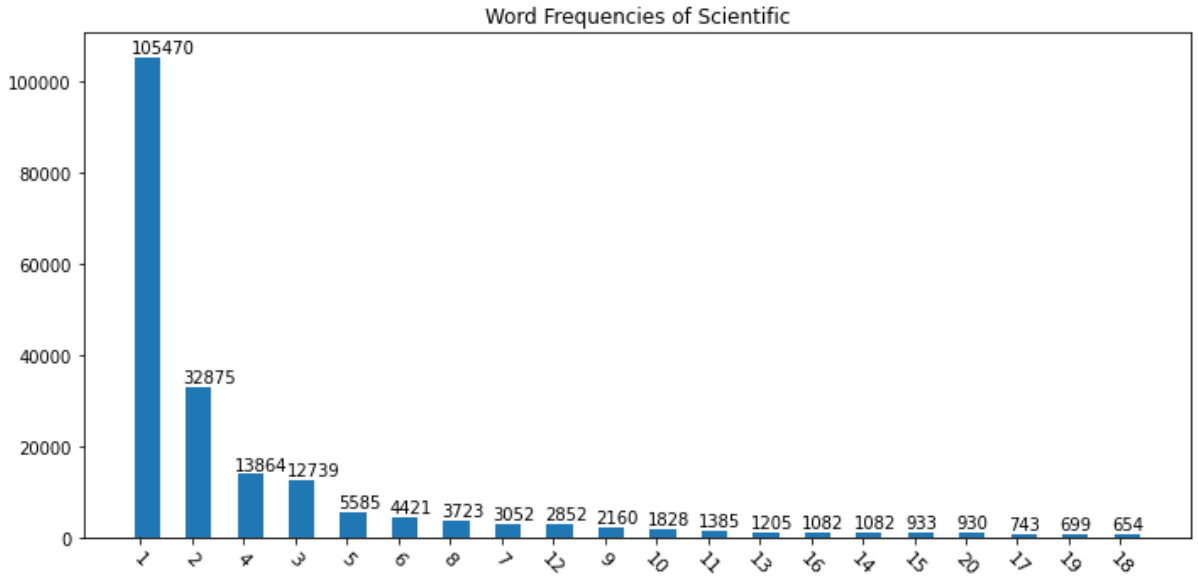

Figura 10: histograma de les freqüències del dataset científic.

A la primera barra que quasi a tots els casos succeirà, tenim que la freqüència 1 que es repeteix per 105470 paraules, o sigui, hi ha 105470 paraules que es repeteixen una vegada, després, hi ha 32875 paraules que es repeteixen 2 vegades, etc.

Naturalment, veiem com s'apropa a una distribució exponencial.

7.3.2.2. Blogs

A la següent figura es mostra l'histograma de freqüències d'aquest dataset.

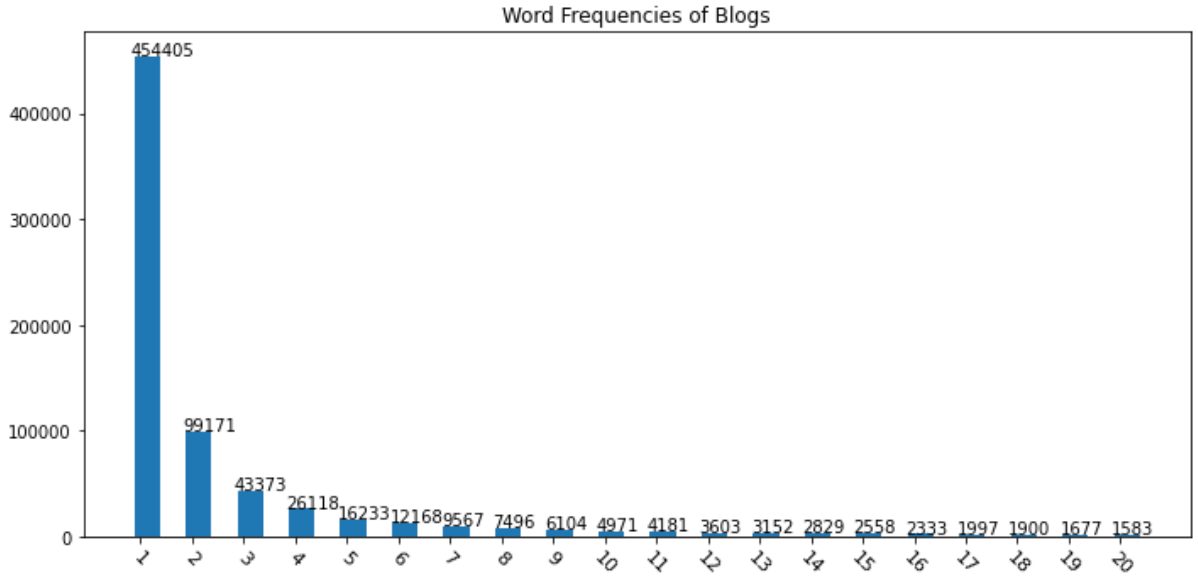

Figura 11: histograma de les freqüències del dataset de blogs.

Una bona diferència entre el dataset científic i aquest és que la distribució de paraules al dataset científic no és lineal respecte el nombre de repeticions i en el de Blogs, els 20 primer sí, essent que és més probable que hi hagi més repeticions de 9 que de 12, això és degut principalment a què el dataset de blogs és bastant més gran que el scientific. Es podria també deduir que al científic en descriure conceptes i assaigs llavors es fa servir grups de paraules molt més sovint relacionats amb el que s'està volguent descriure.

Aquest dataset també s'apropa a una distribució exponencial.

8. Entrenament dels models

Hem vist el preprocessat de documents per tal de poder entrenar correctament els models, en aquesta secció veurem una explicació de com funcionen els models a escala de detall de perceptron, veurem com es desenvolupen les diferents equacions i com es relacionen els diferents paràmetres amb el projecte donant una vista més global dels models.

Aquesta secció està dividida en tres, com funciona word2vec, doc2vec, dins de cada model com es relacionen els seus paràmetres amb la llibreria que fem servir gensim a l'hora d'entrenar el model i fer-lo servir, i finalment com fem servir els models en el projecte per calcular els *embeddings*.

Cal destacar que sempre que entrenem un model el guardem a disc per tal que després, a l'hora d'executar un experiment el puguem carregar d'una manera eficient.

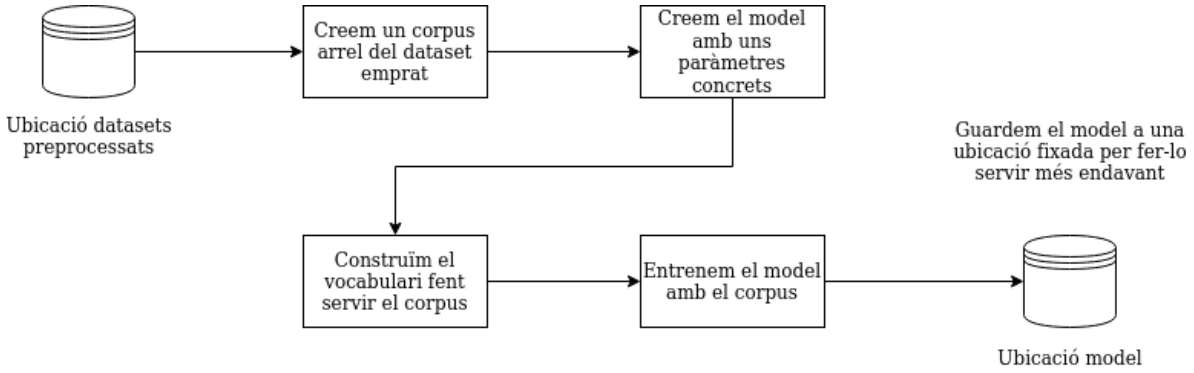

Figura 12: cicle d'entrenament d'un model.

8.1. Word2vec

Word2vec és una estructura de xarxa neuronal específica, on per cada algoritme canvia o bé el nombre d'inputs o bé el nombre d'outputs segons la mida del vocabulari[57] [58].

Assumim que fem servir a tota aquesta secció la one-hot encoding. La mida del vocabulari l'anomenarem **V**.

També assumim que la mida de la "*hidden layer*" és variable respecte a **V**, la denominarem **N**.

No explicarem tota la terminologia que es fa servir al paper original, però sí que veurem de forma intuïtiva com funcionen per dins els models.

[57] "word2vec Parameter Learning Explained - arXiv." https://arxiv.org/pdf/1411.2738. Es va consultar el 14 abr. 2022.
[58] "ronxin/wevi: Word Embedding Visual Inspector - GitHub." https://github.com/ronxin/wevi. Es va consultar el 14 abr. 2022.

8.1.1. One-word context (Bi-gram)

El model més simple de *word2vec* existent és la denominada "*one-word context*" on l'input layer és de la mida del vocabulari **V** i l'output layer és també de la mida del vocabulari **V**.

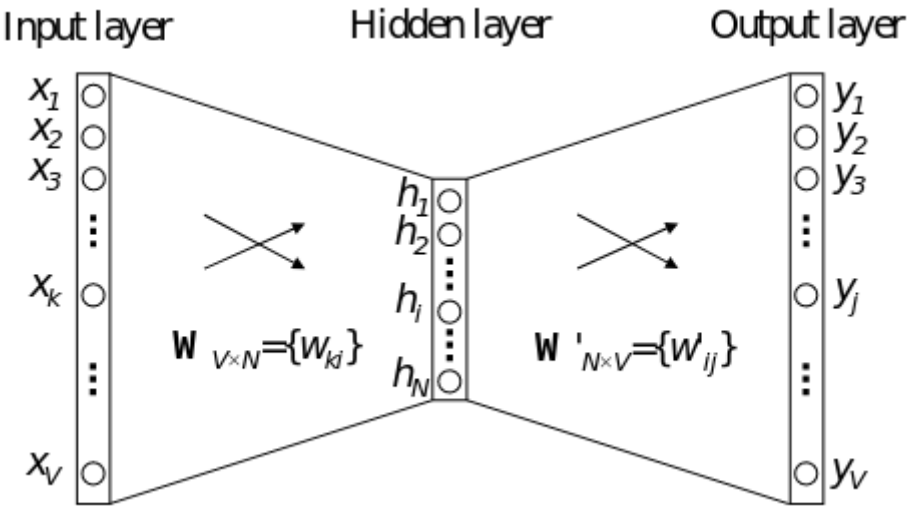

Figura 13: estructura interna de Word2vec en el cas de one word context (bi-gram), extret del paper Xin Rong.

Com veiem a la figura l'input i l'output és la codificació que hem definit anteriorment. Veiem com la comunicació entre l'*input layer* i la *hidden layer* és una matriu amb **V** files i **N** columnes i la comunicació entre la *hidden layer* i l'*output layer* és una matriu de **N** files i **V** columnes. Cada posició d'aquestes dues matrius és un valor associat a un pes, on podem interpretar si aquest "enllaç" ha sigut estimulat o no.

A la figura següent veiem la composició d'una neurona (per exemple, h2 a l'esquema anterior):

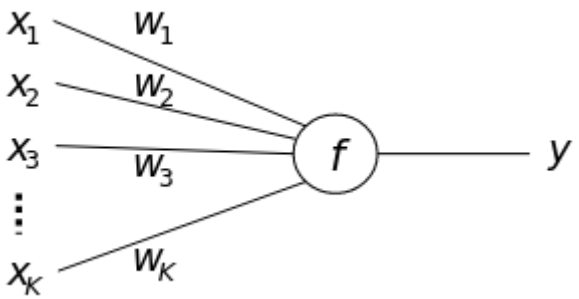

Figura 14: esquema conceptual d'un perceptró, extret del paper Xin Rong.

Per calcular **y**:

$$y = f(u),$$

On **f** és una funció d'activació, i **u**:

$$u = \sum_{i=0}^{K} w_i x_i$$

Figura 15.: funció d'activació d'un perceptró, extret del paper Xin Rong.

Veiem que **u** és una composició de valors formats per *l'input* i els pesos associats a la neurona en qüestió.

Això és aplicable per tota la *hidden layer* i tota l'*output layer*.

Tornant al model, cada vegada que nosaltres volem introduir una nova paraula per tal que el model sigui entrenat existeix una tècnica denominada "*back-propagation*", sense això sempre tindríem el mateix *output* per cada *input* introduït.

Cada pes de les matrius el podem veure com un punt dins d'una dimensió, amb el *backpropagation* el que fem és anar apropant aquest punt a un òptim local, el descens de gradient estocàstic el que fa és ajustar la funció per tal d'anar apropant aquest punt cap aquest òptim local que volem.

Per fer això hem de buscar una funció d'error primer, se li diu també "*training objective*", cap a on ens hem de moure per tal d'arribar a l'òptim local.

Volem tindre normalitzats els pesos, per tant, podem fer servir *softmax* per tal de tindre els pesos entre 0 i 1

$$\sigma : \mathbb{R}^K \to [0,1]^K$$
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{para } j = 1, \ldots, K.$$

Figura 16.: funció softmax, extret de wikipedia "Función Softmax".

Fent servir un model de classificació lineal-logarítmica per obtenir a posteriori la distribució de les paraules, tals que respecta una distribució multinomial

$$p(w_j|w_I) = y_j = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})},$$

Figura 17: distribució multinomial per la distribució de les paraules, extret del paper Xin Rong.

Al model, per tal de computar *"l'score"* de cada paraula al vocabulari fem servir

$$u_j = {\mathbf{v}'_{w_j}}^T \mathbf{h},$$

Figura 18: còmput de l'score de cada paraula del vocabulari, extret del paper Xin Rong.

I la representació de cada paraula *input* donat un context

$$\mathbf{h} = \mathbf{W}^T\mathbf{x} = \mathbf{W}^T_{(k,\cdot)} := \mathbf{v}^T_{w_I},$$

Figura 19: representació de cada paraula input, extret del paper Xin Rong.

Substituint aquestes dues fórmules arribem a

$$p(w_j|w_I) = \frac{\exp\left({\mathbf{v}'_{w_j}}^T \mathbf{v}_{w_I}\right)}{\sum_{j'=1}^{V} \exp\left({\mathbf{v}'_{w_{j'}}}^T \mathbf{v}_{w_I}\right)}$$

Figura 20: equació final de la distribució de les paraules, extret del paper Xin Rong.

$\mathbf{v_w}$ ve de *input->hidden*
$\mathbf{v'_w}$ ve de *hidden->output*

Tenint això ara podem derivar per tal de trobar les equacions corresponents per tal d'anar actualitzant els pesos.

8.1.1.1. Equació per hidden->output

Volem maximitzar l'anterior fórmula, $\mathbf{w_O}$ és *l'ouput word* i $\mathbf{w_I}$ és l'*input word*. **j*** denota l'índex a l'*output layer*.

$$\begin{aligned} \max p(w_O|w_I) &= \max y_{j^*} \\ &= \max \log y_{j^*} \\ &= u_{j^*} - \log \sum_{j'=1}^{V} \exp(u_{j'}) := -E, \end{aligned}$$

Figura 21: maximització de la probabilitat de l'output word respecte a l'input word, extret del paper Xin Rong.

És equivalent a minimitzar **E**, és la nostra *loss function* i **j*** és l'índex de l'actual *output word* a la *output layer*. Derivem **E** respecte $\mathbf{u_j}$:

$$\frac{\partial E}{\partial u_j} = y_j - t_j := e_j$$

Figura 22: derivada de la loss function respecte a l'entrada neta de la unitat, extret del paper Xin Rong.

$$t_j = \mathbb{1}(j = j^*), \text{ i.e., } t_j$$

Figura 23: assignació d'u o zero.

Només serà 1 quan la unitat **j** és l'actual output word, altrament $\mathbf{t_j = 0}$

Per tant, volem derivar $\mathbf{w'_{ij}}$ per tal de poder calcular la matriu corresponent:

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial w'_{ij}} = e_j \cdot h_i$$

Figura 24: derivació de E respecte a cada posició de la matriu, extret del paper Xin Rong.

Finalment, aplicant descens de gradient estocàstic obtenim l'equació pels pesos *hidden-output*:

$$w'^{(\text{new})}_{ij} = w'^{(\text{old})}_{ij} - \eta \cdot e_j \cdot h_i.$$

Figura 25: equació dels pesos hidden-output, extret del paper Xin Rong.

Aquesta equació surt del *learning algorithm* del perceptró.

8.1.1.2. Equació input->hidden

Per trobar-la fem servir la *loss function*, derivem aquesta respecte a la *hidden layer*:

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} := \mathrm{EH}_i$$

Figura 26: derivació de la loss function respecte a la hidden layer, extret del paper Xin Rong.

$\mathbf{u_j}$ és definit a les primeres equacions.

Volem la derivada de **E** respecte a cada element de la matriu **W**:

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial w_{ki}} = \mathrm{EH}_i \cdot x_k$$

Figura 27: derivació de E respecte a la matriu W, extret del paper Xin Rong.

Això és equivalent al producte tensor de **x** i **EH**:

$$\frac{\partial E}{\partial \mathbf{W}} = \mathbf{x} \otimes \mathrm{EH} = \mathbf{x}\mathrm{EH}^T$$

Figura 28: equivalència amb el producte tensor, extret del paper Xin Rong.

D'aquí obtenim una matriu **V x N**, com que només un dels components de **x** és diferent de zero, el valor de la fila és $\mathbf{EH^T}$ i, per tant:

$$\mathbf{v}_{w_I}^{(\text{new})} = \mathbf{v}_{w_I}^{(\text{old})} - \eta \mathrm{EH}^T$$

Figura 29: equació dels pesos input-hidden, extret del paper Xin Rong.

Amb aquestes dues equacions podem fer el *backpropagation* per tal de poder entrenar el nostre model.

Amb això tenim l'esquema complet per començar a entrenar el nostre model, codifiquem a *one-hot*, introduïm al model, calculem l'*output*, calculem l'error i apliquem la fórmula.

8.1.2. Multi-word context (Continuous Bag of Words)

Hem explicat com funciona per una paraula d'*input* i una paraula d'*output*, cal explicar com funciona amb múltiples paraules d'*input*.

Aquest model s'entrena de manera que introdueixes les **C** primeres paraules i ha de predir la **C + 1** paraula, calculem l'error i fem *backpropagation*.

Seguim amb la mateixa codificació, però afegint més paraules d'entrada, això vol dir que si les nostres paraules tenen una mida **V** llavors l'*input,* serà **CxV** amb **C** nombre de paraules d'entrada.

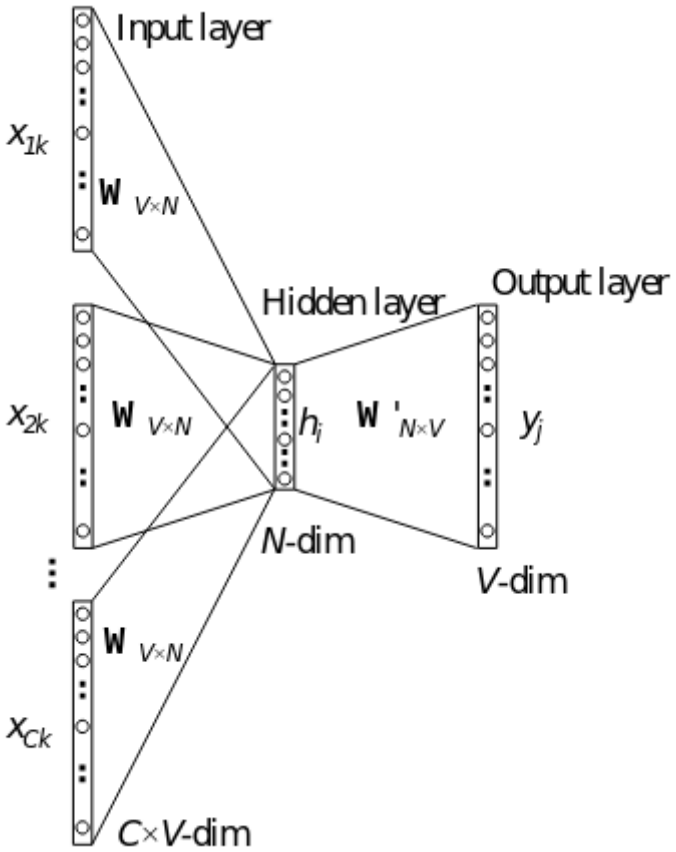

Figura 30: estructura interna de Word2vec en el cas de multi word context (Continuous Bag of Words), extret del paper Xin Rong.

I, per tant, la matriu *input->hidden* que tenim ara:

$$\begin{aligned}\mathbf{h} &= \frac{1}{C}\mathbf{W}^T(\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_C) \\ &= \frac{1}{C}(\mathbf{v}_{w_1} + \mathbf{v}_{w_2} + \cdots + \mathbf{v}_{w_C})^T\end{aligned}$$

Figura 31: matriu input-hidden actual, extret del paper Xin Rong.

On $\{\mathbf{w_1}, \ldots, \mathbf{w_C}\}$ són les paraules del context i $\mathbf{v_w}$ és el vector *input* de la paraula **w**. Definim la *loss function*:

$$\begin{aligned} E &= -\log p(w_O|w_{I,1}, \cdots, w_{I,C}) \\ &= -u_{j^*} + \log \sum_{j'=1}^{V} \exp(u_{j'}) \\ &= -\mathbf{v'_{w_O}}^T \cdot \mathbf{h} + \log \sum_{j'=1}^{V} \exp(\mathbf{v'_{w_j}}^T \cdot \mathbf{h}) \end{aligned}$$

Figura 32: càlcul de la loss function per CBOW, extret del paper Xin Rong.

És una adaptació de la primera equació del pas "*Equació per hidden->output*", podem veure que només hem substituït, ja que només hem canviat les dimensions de la matriu *input->hidden*.

Les equacions d'actualització *hidden->ouput* es queden igual i les equacions d'actualització per *input->hidden* la qual cosa ara hem d'aplicar no només per una paraula sinó per cada paraula de l'*input* la següent equació:

$$\mathbf{v}_{w_{I,c}}^{(\text{new})} = \mathbf{v}_{w_{I,c}}^{(\text{old})} - \frac{1}{C} \cdot \eta \cdot \mathrm{EH}^T \qquad \text{for } c = 1, 2, \cdots, C.$$

Figura 33: equació input-hidden per CBOW, extret del paper Xin Rong.

On $\mathbf{v_{wI,c}}$ és l'*input* vector de la paraula **c-éssima** de *l'input*.

No s'ha hagut de canviar molt la fórmula, ja que **E** la teníem aïllada i la podem redefinir amb el que hem fet abans.

8.1.3. Skip-gram Model

Aquest model consisteix a deixar una paraula com a *input* i **n** paraules com a *output*.

La manera d'entrenar aquest model és inversa, o sigui, s'introdueix la primera paraula del document i ha de predir les **n** següents paraules, calculem l'error i fem *backpropagation*.

És més acurat per paraules menys freqüents, però molt més lent (fer el càlcul de les equacions de backpropagation és molt costós)

El model Skip-gram no s'explicarà en detall perquè queda fora de l'abast del projecte, però un esquema és:

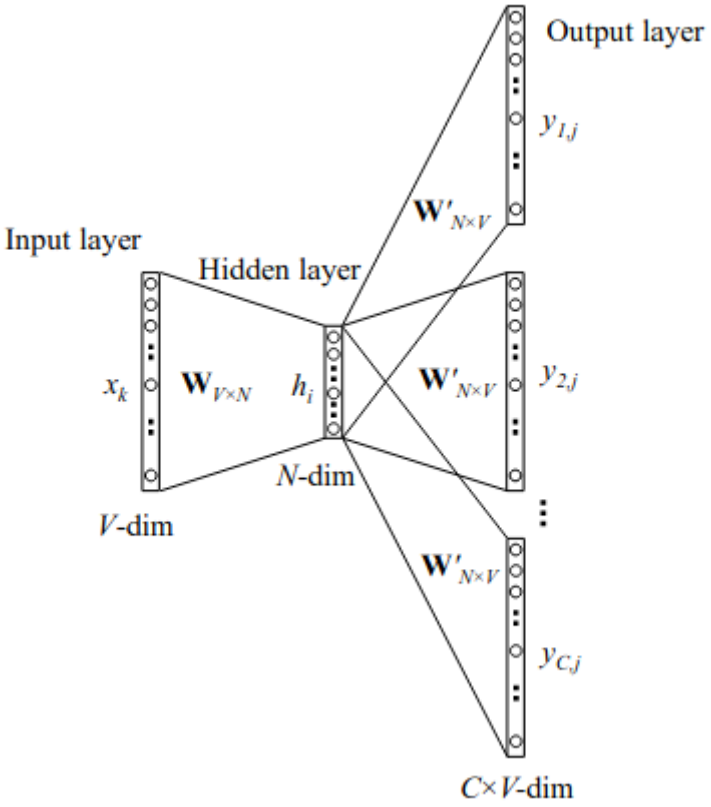

Figura 34: estructura interna de Word2vec en el cas de multi word context (Skip-Gram), extret del paper Xin Rong. .

Existeixen dos tipus de models a *word2vec*, *Continuous Bag of Words* i *Skip-Gra*m.

Dins aquest concepte existeixen dos algoritmes per donar forma a aquestes representacions:

8.1.4. Aplicació al projecte

La llibreria gensim porta uns paquets que permeten fer servir *Word2Vec*, hi ha diversos paràmetres configurables, d'entre altres, ens interessen els següents:

8.1.4.1. size

Dimensionalitat dels vectors de paraules, afecta la velocitat d'entrenament i a la precisió del model final. No confondre amb el paràmetre **V** de les equacions, aquest paràmetre *size* fa referència a l'hora de calcular els *embeddings*.

8.1.4.2. window

Màxima distància entre la paraula actual i la paraula següent dins una sentència. És el paràmetre **C** de les equacions, evidentment contra més gran més precisió, però alhora el cost d'entrenament del model es fa molt gran.

8.1.4.3. max_vocab_size

Aquest sí és el paràmetre **V**, tantes dimensions com paraules al vocabulari, com que el vocabulari no es dóna en el moment de crear el model es pot definir un màxim i anar assignant paraules a cada dimensió.

8.1.4.4. sg

Indica quin dels dos algorismes es fa servir per entrenar el model. O bé *Continuous Bag-of-Words* o bé *Skip-gram*.

8.2. Doc2vec

La diferència principal amb *Word2vec* és que s'introdueixen diversos vectors com a input a part de les paraules de la finestra. Aquests vectors es poden codificar d'una manera diferent de les altres paraules. Veiem doncs els dos tipus d'algorismes[59]:

8.2.1. Distributed Memory version of Paragraph Vector (PV-DM)

Aquest tipus de model és equivalent a *Continuous Bag-of-Words*. La manera de calcular les equacions de backpropagation i les equacions d'*output* és de la mateixa, però simplement afegint un vector més a l'*input*, per tant, sempre tindrem **C + 1** paraules com a *input*.

El vector afegit s'anomena el *label* de document o paràgraf, és una paraula que caracteritza el document o paràgraf en qüestió, pot ser de mida diferent dels altres vectors, ja que parlem d'un altre tipus de grup de paraules. Evidentment, si fem servir una codificació diferent per aquest nou label, s'hauran de recalcular moltes de les fórmules que s'han explicat a *Word2vec*.

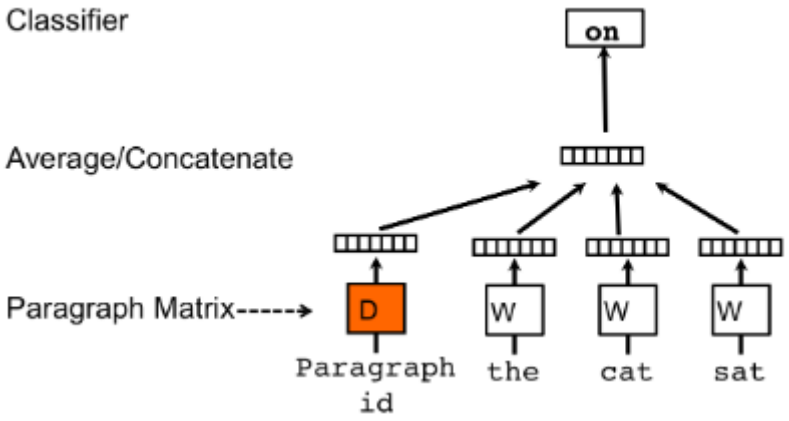

Figura 35: estructura interna de Doc2Vec al cas de PV-DM, extret del paper de Quic Le i Tomas Mikolov .

Podem arribar a definir més d'un vector paràgraf, depèn el context que vulguem definir al nostre model, per exemple, amb etiquetes que vagin variant per fitxer, per document i per paràgraf.

Com veiem a la figura, tenim un vector "*Average/Concatenate*" que és una representació del pas final d'*output* (*hidden layer* + *output layer*).

[59] "Distributed Representations of Sentences and Documents - Stanford ...." https://cs.stanford.edu/~quoc/paragraph_vector.pdf. Es va consultar el 14 abr. 2022.

8.2.2. Distributed Bag of Words version of Paragraph Vector (PV-DBOW)

Idea similar a *Skip Gram*, però el vector o vectors que fem servir són els ja comentats vectors de paràgraf i no es fa servir com a input cap paraula del context, simplement mitjançant una etiqueta de paràgraf podem predir les **C** paraules de la finestra.

Existeixen dues versions, una on no s'emmagatzemen els vectors finals i altre que sí, al paper només es referència la primera, però hem vist que a *Skip-Gram* es fa servir la segona.

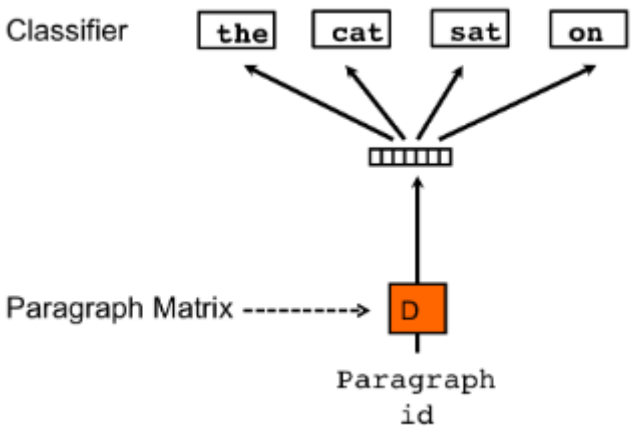

Figura 36: estructura interna de Doc2Vec al cas de PV-DBOW, extret del paper de Quic Le i Tomas Mikolov

8.2.2.1. Versió sense emmagatzemar

L'*output layer* té una mida de **V** dimensions a un vector, tantes com el vocabulari, la finestra és **C**, amb **C < V**, per tant, en aquest vector només tindrem **C** uns i **V - C** zeros. Aquesta versió, per tant, té un cost lleuger a memòria perquè no ho hem d'emmagatzemar tot. Es veu clarament que no es respecta el one hot encoding al vector de sortida.

La manera d'entrenar aquest model és anar fitant el nombre de paraules que hauria de ficar com a uns al vector, és a dir, definir una finestra de **C** paraules, i amb el backpropagation anar corregint els desviaments.

8.2.2.2. Versió emmagatzemant

L'*output layer* té **C** vectors de **V** dimensions, on cada vector és un *one hot encoding*. És idèntica a *Skip-Gram* i, per tant, l'*output* és gran i s'ha d'anar calculant.

8.2.3. Aplicació al projecte

Dins la llibreria de gensim podem trobar un paquet dedicat a *Doc2Vec*, podem configurar entre molts, els següents paràmetres que corresponen molt amb els escollits a *Word2vec*:

8.2.3.1. documents
Tots els identificadors de documents (etiquetes) que es faran servir. Està relacionat amb el paràmetre de vectors de paràgrafs comentats anteriorment, es crea a priori la codificació que es farà servir per aquests vectors.

8.2.3.2. vector_size
Aquí es defineix la dimensionalitat dels vectors de paraules, el paràmetre **V**, és a dir, la mida del vocabulari.

8.2.3.3. window
El nombre de paraules del context, a les descripcions li hem anomenat **C**.

8.2.3.4. dm
Defineix l'algorisme d'entrenament, es pot escollir entre els dos esmentats, 1 per *PV-DBOW* ó 0 per *PV-DM*.

8.3. Utilització dels models

Tots els models generats els hem guardat a disc, la intenció és que cada vegada que vulguem executar una comparació de documents el puguem carregar de disc ja entrenat.

La mateixa llibreria de gensim ens permet exportar els models en un format binari i carregar-los com a un objecte amb una interfície per tal de poder-lo fer servir.

8.3.1. Càlcul d'embeddings

La pedra angular del projecte és poder calcular un *embedding* d'un document per tal de poder comparar-lo d'alguna manera amb un d'altre.

Un embedding d'un document és la representació vectorial d'aquest document en un espai euclidià a $\mathbf{R^n}$ amb **n** nombre de dimensions.

Existeixen diverses maneres de representar un embedding:

- **TfIdf**: explicat a un dels primers apartats del document, mitjançant un *corpus* podem obtenir una sèrie de pesos els quals es poden comparar arran d'un altre document calculat de la mateixa manera.
- **Matrius de paraules**: des de *Word2Vec* i *Doc2Vec* es pot accedir directament a la matriu de paraules una vegada estan els models a memòria.

Per cada tipus de model es calcularà de manera diferent un *embedding*, s'ha de tindre en compte que cada arquitectura té les seves particularitats i els seus conceptes els quals s'han de tindre en compte.

Aquí presentarem embeddings per matrius de paraules extretes de *Word2Vec* i *Doc2Vec*.

Cada paraula del document es pot localitzar a l'espai euclidià representat pel propi model, per tant, si transformem el vector de paraules a un vector de vectors en aquest espai podrem fer la mitja de tots per veure el punt del document en aquest espai.

Tenint aquesta mitja podrem comparar en parells de documents la distància entre ells.

9. Aplicació de mètriques

Hem vist com es preprocessen els documents i es guarden, després hem vist com fem servir aquests documents per entrenar els models i hem donat una perspectiva més en profunditat de com funcionen els models, ara el que explicarem en aquesta secció és com fem servir tot això per poder crear experiments d'una manera ràpida i simple.

En aquesta secció tenim la rutina principal, és una rutina parametritzable per tal d'executar tot el projecte combinant diversos paràmetres com el model, el context, etc.

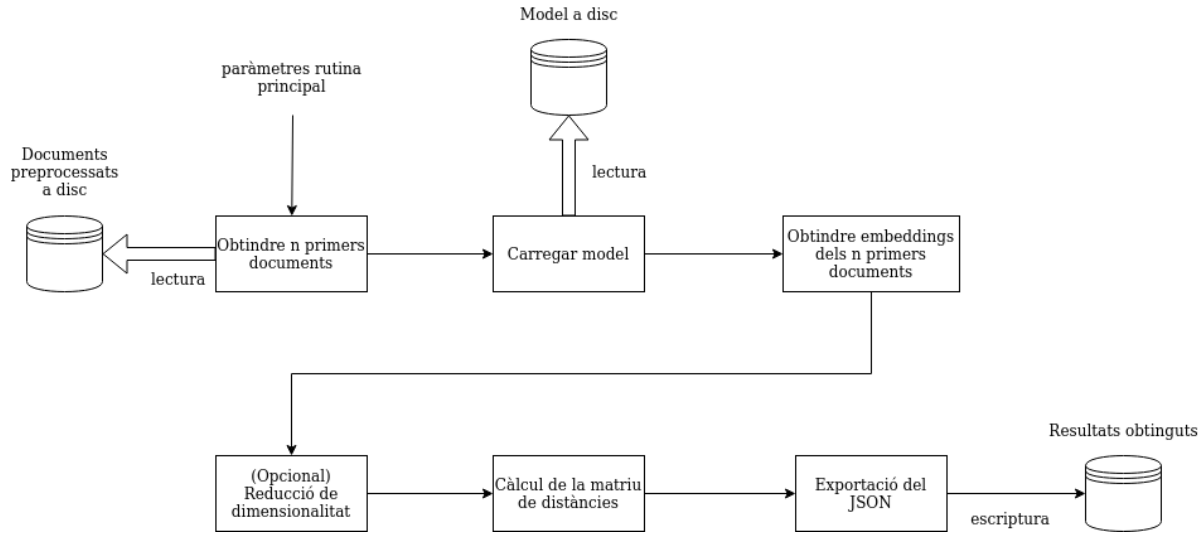

Figura 37: pipeline del funcionament de la rutina principal.

Aquesta rutina es compon de diferents fases, segons els paràmetres donats aplicarà a cada pas una cosa o un altre. Per exemple, la càrrega del model depèn del context sobre què volem executar la rutina i el tipus de model que indiquem (*Doc2Vec* o *Word2Vec*).

Es detalla cada pas amb més detall a aquesta secció i les diferents particularitats de la implementació que s'ha fet servir.

9.1. Rutina principal

Aquesta és la rutina principal d'experimentació, s'ha creat de manera que sigui el més flexible possible per tal de poder crear diferents combinacions de paràmetres i executar-los dins d'un bucle, cada execució genera un arxiu amb els resultats d'aquesta execució, el nom que se li dóna a aquest és la combinació de paràmetres i la data.

Aquesta rutina admet com a paràmetres:

- **Mètode**: quin tipus de prova volem aplicar, si per documents o per classes de documents.
- **Tipus de model**: el model que farem servir, *Word2Vec* o *Doc2Vec*, s'havia plantejat més cap al final de projecte implementar proves per els dos tipus d'algorismes amb *Word2Vec* i amb els dos tipus d'algorismes amb *Doc2Vec*.
- **Mètrica**: quina mètrica farem servir per fer les comparacions, comentades al principi del document.
- **Context**: quin context farem servir per fer les comparacions, al principi del document s'ha detallat aquest aspecte.
- **Carpeta destí**: és el directori on es deixaran tots els resultats d'aquesta execució.
- **Nombre de documents**: sempre s'agafaran els mateixos documents per ordre, o sigui, si s'indica 5, s'agafaran sempre els 5 primers documents.
- **Paràmetre extra**: la mètrica *Minkowski* rep un hiperparàmetre per fer el càlcul.
- **Activació *TSNE***: fins bastant tard del projecte no es va donar l'opció que sempre es faci servir aquest mètode abans de calcular la mètrica, i és donat que es va introduir la mètrica *Haversine* que exclusivament es pot fer amb dues dimensions, per tal que totes les comparacions siguin versemblants es va introduir aquesta reducció de dimensionalitat.
- **Components**: relacionat amb el *TSNE*, indica la dimensionalitat de l'*output*, nosaltres farem servir 2.
- ***Perplexity***: hiperparàmetre del *TSNE*, explicat a la secció del document corresponent.

Tots aquests paràmetres ens permeten coordinar proves d'una manera bastant ràpida.

Dins d'aquesta rutina tenim diverses rutines:

9.1.1. Obtenir n primers documents

En aquesta rutina s'agafen documents dels dos tipus de contextos, blogs i científic.

És important recalcar que aquests documents són els preprocessats anteriorment a la *pipeline* de preprocessament que s'explica a l'apartat corresponent del document.

Per tal de llegir els documents, es llisten els n primers fitxers de text de les dues carpetes on hi són emmagatzemats i es llegeix fitxer per fitxer el contingut, al següent pas es talla la cadena de text resultant per espais i s'assoleix un llistat de paraules, a l'estar preprocessats tenim una string amb paraules separades per espais únicament. Cada llistat s'etiqueta amb el nom del document.

9.1.2. Obtenir embeddings dels documents

Inclou els selectors pel context i el tipus de model, el resultat és una matriu. En aquest apartat es dóna una funció que transforma un llistat de paraules (document) a un llistat de pesos (*embedding*).

Com veiem, aquesta implementació està amagada i aquí a l'hora de generar la funció és quan es carrega el model. Més endavant aquesta funció s'anirà propagant per les següents rutines.

Aquesta funció es podria fer també en forma de classe amb alguna mena d'encapsulació del model, encara que per un projecte d'aquesta mida no s'ha cregut necessari.

Abans d'obtenir cada *embedding* corresponent es filtren les paraules que no hi siguin al vocabulari, ja que d'altra manera és impossible calcular l'*embedding*, recordem que si no existeix una codificació d'una paraula a *l'input* de la xarxa neuronal, és impossible ubicar aquesta dins de l'espai euclidià.

9.1.2.1. Càlcul d'embedding Word2Vec

Al càlcul de l'*embedding* a *Word2Vec* es calcula com es comenta a la secció anterior, entrarem en detalls més tècnics.

El model té emmagatzemat el vocabulari, per tant, primer, filtrem les paraules que no hi són al vocabulari. Després aconseguim els pesos corresponents a les paraules que tenim al document derivat del vocabulari, per cada paraula aconseguim una llista de valors que corresponen a les coordenades en **n** dimensions, si es repeteix qualsevol paraula ha de comptar també com una paraula (no es fa una funció per distingir paraules dins el document), això té com a resultat una matriu de **x** paraules (tantes com tingui el document) per **n** dimensions.

Una vegada tenim la matriu sumem els vectors, però ha de ser una suma per coordenades, és a dir, si en tinc dos punts **(x1, x2)** i **(y1, y2)**, el resultant ha de ser **(x1+y1, x2+y2)** i després cada coordenada l'hem de dividir pel nombre de paraules pel fet que les sumes realitzades a cada coordenada és de l'ordre del nombre de paraules del document. Així doncs, obtenim la mitja de totes les coordenades de tot el document resultant així en una llista de valors que representa el punt a l'espai euclidià del document a **n** dimensions.

9.1.2.2. Càlcul d'embedding amb Doc2Vec

És molt més simple que amb *Word2Vec* ja que el model de *Doc2Vec* ens permet fer servir un mètode anomenat "*infer_vectors*" que ens fa tot aquest treball. Aquest mètode ens troba el punt comentat a l'anterior secció. Evidentment, s'ha de filtrar abans també el document amb el vocabulari del document.

9.1.3. Calcular matriu de distàncies

Aquest pas inclou la reducció de dimensionalitat i el càlcul de la mètrica.

La reducció de dimensionalitat l'hem fet amb *TSNE* i es fa servir a tota mètrica, ja que no tota mètrica compleix el requisit de funcionar a **n** dimensions, a una secció anterior s'explica en detall.

Aquesta funció de reducció de dimensionalitat rep com a paràmetres els hiperparàmetres de *TSNE* i una funció tal que de dos valors transforma a un, en aquest cas és la signatura d'una funció de mètrica tal que de dues llistes de punts en calcula la distància entre aquestes, cada llista de punts representa un document i el resultat ha de ser un nombre real que representarà la distància.

A l'hora de parametritzar el *TSNE* podem configurar la mètrica que fa servir internament per fer la reducció de la dimensionalitat. En aquest cas hem fet servir també la mètrica donada per paràmetre per tal que s'ajusti el màxim possible el càlcul global a l'assaig que s'està fent en aquell moment.

Una vegada feta la reducció de dimensionalitat, calculem la distància entre cadascun d'aquests un a un, si tenim **n** documents es donaran **n^2** càlculs i la quantitat de resultats serà el mateix. Cada resultat tindrà assignats dues etiquetes que seran el nom dels dos documents comparats.

Falta normalitzar el resultat, es fa aquesta operació perquè diferents mètriques interpreten els punts de manera diferent i fan que es desajusti la interpretació del resultat, perquè per exemple, poden haver mètriques que el resultat sigui de 0.0001 i altres de 5879; i no necessàriament una és molt petita i altre molt gran a efectes de la mètrica, vol dir que la del 0.0001 altres càlculs en aquesta mètrica donen resultats propers, ídem amb 5879 i a l'hora de donar una interpretació podria semblar que estan molt lluny.

Per tant, el procés de normalització inclou, agafar el màxim i el mínim, el mínim sempre serà 0, ja que incloem mesura de distància entre els mateixos documents (la diagonal de la matriu de comparació resultant) i el màxim depèn la mètrica. Així doncs, per cada distància mesurada se li resta el mínim i es divideix entre la diferència del màxim i del mínim, aquest càlcul sempre dona un resultat entre 0 i 1, fent que sigui estàndard a tota interpretació entre diferents mètriques.

Hem volgut amagar la implementació d'aquest mètode dins d'una funció tal que com a paràmetre es dóna un llistat d'etiquetes juntament amb el seu llistat de punts que representen el document a l'espai euclidià; com a resultat es dóna una matriu **n^2** amb dues etiquetes i la seva distància calculada.

9.1.4. Calcular i exportar

En aquesta rutina és on es fa tot el treball computacional, tot el anterior només han sigut definicions i encapsulaments per tal de fer-ho servir més endavant.

Tenim diversos paràmetres, però són referents a què s'està fent servir en forma de *string*, no té més utilitat que donar-li un nom final a l'arxiu exportat i poder saber com s'ha calculat aquella matriu de distàncies.

Hi ha dos apartats, càlcul i exportació:

9.1.4.1. Càlcul

Inclou el càlcul de l'*embedding* del document, la matriu de distàncies i si cal, la mitjana entre classes de document (això s'explicarà més endavant, ja que és un mètode d'experimentació).

Com veiem, les operacions de calcular l'*embedding* del document i la matriu de distàncies derivada del càlcul de *l'embedding* són definides per funcions ja creades anteriorment.

Els passos que es fan és, primer, obtenir els *embeddings*, iterem per cada document donat i calculem el seu *embedding*.

Una vegada tenim *l'embedding* podem fer servir la funció del càlcul de la matriu de distàncies. I fet això ens queda veure si volem agrupar per classes o no.

9.1.4.2. Exportació

L'exportació consta de dos passos, el primer és transformar la matriu de distàncies aconseguida a un format *JSON* i el segon és el d'exportar a un fitxer aquest *JSON*.

És important recalcar que l'ordre dels documents dins de la matriu és el mateix que l'ordre de lectura de documents del primer pas de tota la pipeline de processament, per definició de la rutina de càrrega de documents els documents del context científic sempre es carregaran abans que els de blogs, això vol dir que l'exportació de documents, el primer quadrant sempre correspondrà a distàncies mesurades entre científic i científic, el segon i tercer entre científic i blogs i el quart entre blogs i blogs, això ens pot ajudar a veure ràpidament les relacions entre distàncies de diferents contextos.

9.1.4.2.1. Matrix to JSON

El format com a *input* de la matriu de similaritat és de *List<Pair<string,List<Pair<string,float>>>>*, això vol dir que tenim per cada distància dos documents associats. Volem que a l'hora d'exportar mantinguem aquesta estructura i a més, poder mantenir informació rellevant de l'experiment fet.

Descrivim el format d'aquest JSON en forma de tipus de typescript que és molt clar.

```
type MatrixJson = {
        method: string,
        metric: string,
        model: string,
        context: string,
        components: string,
        perplexity: string,
        list: {
                name:string,
                vecs: {
                        name: string,
                        value: float
                }[]
        }[]
}
```

Un exemple d'aquesta estructura pot ser:

```
{
        method:"nxnrows",
        metric: "Euclidean",
        model: "doc2vec",
        context: "scientific",
        components: "2",
        perplexity: "4",
        list: [
                {
                        name: "Doc1",
                        vecs: [
                                {
                                        name: "Doc1",
                                        value: 0.0
                                },
                                {
                                        name: "Doc2",
                                        value: 0.8
                                },
                                …
                                {
                                        name: "DocN",
```

```
                                        value: 0.1
                                }
                        ]
                },
                …
                {
                        name: “DocN”,
                        vecs: [
                                {
                                        name: “Doc1”,
                                        value: 0.1
                                },
                                …
                                {
                                        name: “DocN”,
                                        value: 0.0
                                }
                        ]
                }
        ]
}
```

Hi ha dues coses a destacar, la informació addicional que s’ha afegit i les distàncies.

La informació addicional que s’ha afegit ens serveix després per poder discriminar diferents experiments i treure conclusions, per exemple, si hem fet un experiment per tota combinació d’opcions, poder fer que ens agrupi per mètrica les distàncies i comparar en termes generals com es comporten. Es pot fer per mètrica, per context, per model, etc; o inclús crear combinacions de discriminacions.

La segona part és una estructura que ens facilita l’accés per poder representar en matrius de similaritat visuals amb intensitats de colors diferents per tal de representar com de lluny o a prop està cada document, al tindre-ho ja preparat per tractar-ho com una matriu directament és molt més simple fer aquest gràfic.

9.1.4.2.2. JSON to file

Donada una carpeta construïm un nom pel fitxer depenent els seus paràmetres inicials. Una execució combinatòria generarà diversos fitxers, volem que en aquesta prova tot s’agrupi en una sola carpeta, aquesta ve donada sempre per un paràmetre a la rutina principal, el nom s’autogenera depenent els paràmetres amb els quals s’ha executat la prova, fent així que tinguem cada assaig ben classificat.

10. Visualització de resultats

A aquesta secció veurem diferents tipus de visualitzacions que s'han fet servir al projecte, com s'han creat i per què a s'ha fet servir. Més endavant veurem que encara són insuficients pels últims experiments, ja que necessitem dades encara més agregades com gràfics de barres, no es presenten aquí, ja que no necessiten cap justificació a causa de la simplicitat d'aquests.

10.1. Tipus de gràfics

En aquest apartat veurem els tipus de gràfics que s'han anat creant per tal de representar la informació, a mesura que hem anat avançant al projecte hem deixat de fer servir alguns i fent servir altres, només s'explica el tipus de gràfics, no el contingut, el contingut s'explicarà més endavant.

10.1.1. D3 Matriu n documents

Aquesta matriu està feta en *D3* i és una primera versió de la matriu de **NxN** documents, l'objectiu és poder visualitzar ràpidament el comportament dels documents exposats i poder veure un a un perquè són semblants per exemple. Es va descartar aquesta visualització perquè automatitzar-ho per comparar grans quantitats de documents és complicat a l'estar en angular el projecte.

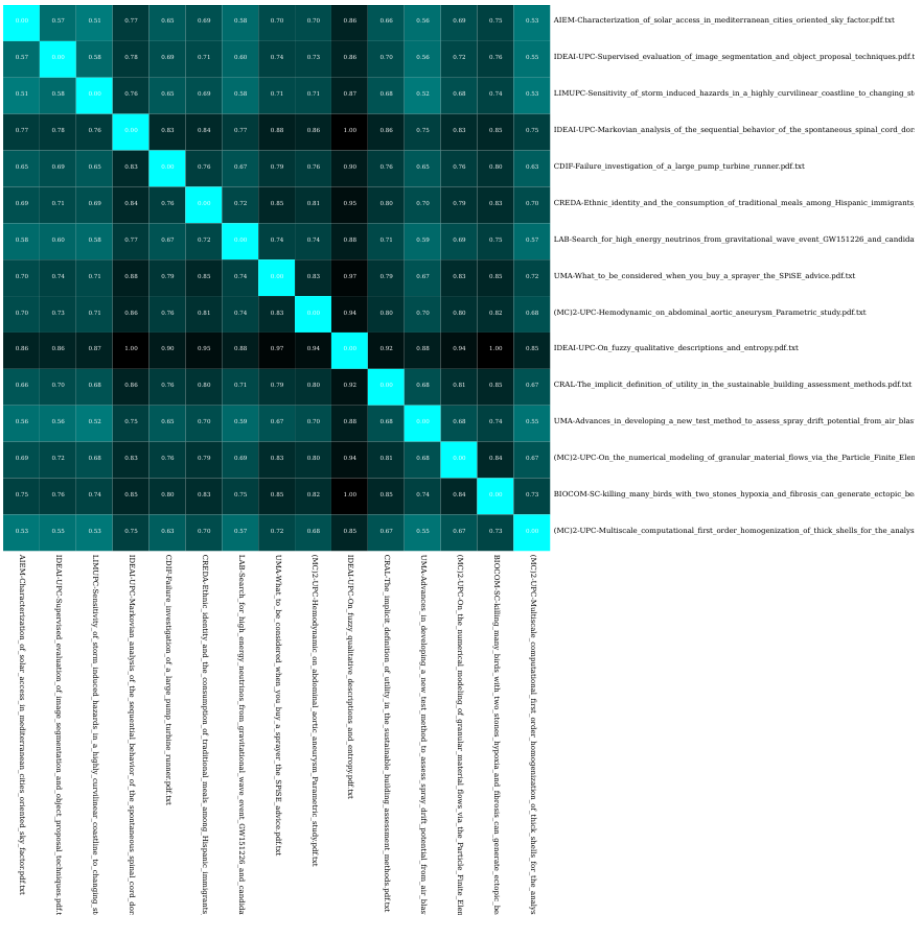

Figura 38: exemple matriu similaritat D3 per n Documents.

10.1.2. D3 matriu by classes

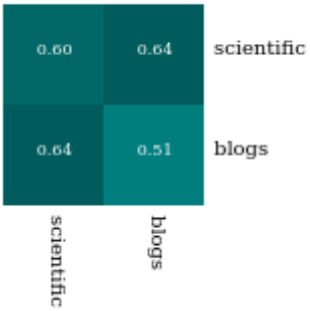

Figura 39: exemple matriu similaritat D3 per classes.

Ídem que amb la matriu de **n** documents. En aquest cas només són dues classes, però amb exactament el mateix format, simplement la matriu és de **2x2** sempre.

10.1.3. D3 Plot 2 dimensional scatter (núvol de punts)

El scatter el fem servir per representar punts en dues dimensions, està implementada una funció per poder incrementar o decrementar la mida dels punts o bé la mida del text en temps real. Es va crear per les primeres visualitzacions dels models per tal de veure com es comportaven els models.

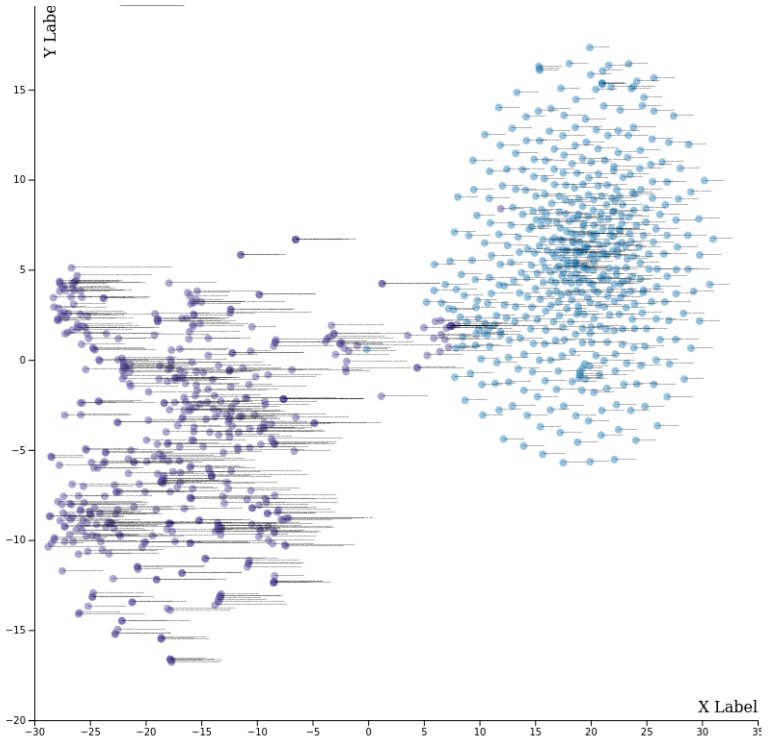

Figura 40: exemple gràfic de núvol de punts en dues dimensions amb D3.

10.1.4. Matplotlib Matriu nxn documents

És una imatge generada automàticament per tal de poder generar grans quantitats de matrius de similaritat. Els resultats eren molts i va ser convenient fer un procés automatitzat per tal de fer-ho. Es detalla més endavant com s'ha fet.

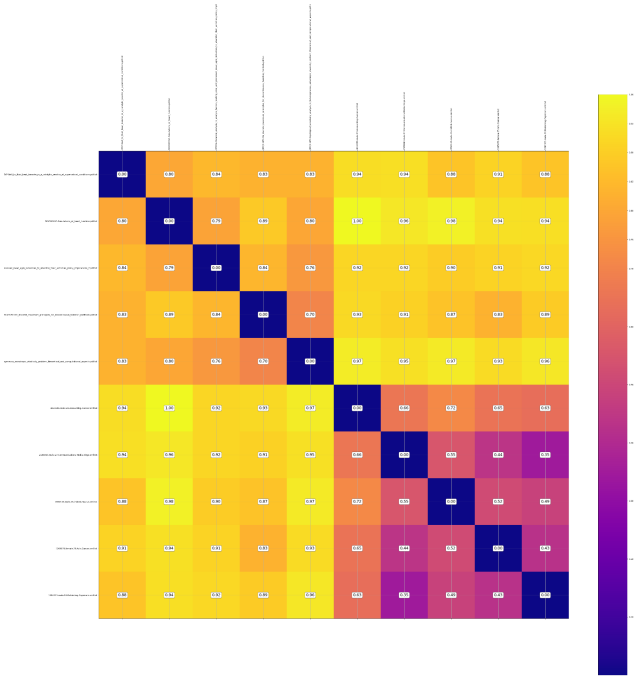

Figura 41: exemple matriu similaritat feta amb Matplotlib per nxn documents.

10.1.5. Matplotlib Matriu per classes

Ídem que amb l'anterior però amb classes.

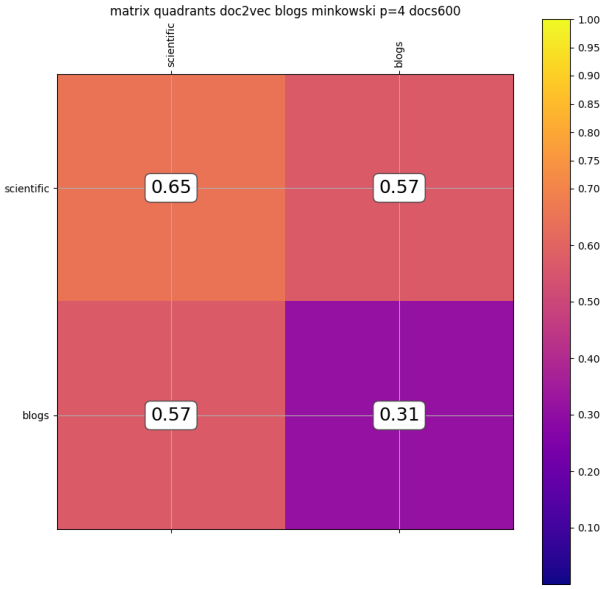

Figura 42: exemple matriu similaritat feta amb Matplotlib per classes.

10.2. Justificacions visualització

En aquest apartat es detalla la transició de *D3* a gràfics de *python* en les matrius de similaritat.

10.2.1. Matriu similaritat D3

Una primera versió es va fer amb *D3* i *Angular* construint una matriu de similaritat en gràfics web carregant manualment un fitxer generat pel projecte de *python*.

A angular accedir als fitxers de l'ordinador requereix d'ordres manuals per l'usuari, com a molt es poden arribar a carregar diversos fitxers de cop, però si es vol anar a un nivell d'automatització superior, és impossible fer-ho, quan hem de comparar moltes matrius de similaritat és molt carregós, per tant, es va idear un segon mètode per avaluar resultats de proves.

10.2.2. Matriu similaritat en Python

Les característiques del projecte fan que l'*output* sigui de centenars de fitxers de resultats, en conseqüència, els requeriments que tenim són:

- Visualització ràpida de matrius de similaritat (necessàriament ha de ser un objecte visual i amb colors).
- Que sigui ràpid canviar de matriu a matriu (ideal una imatge a una galeria).
- Que la generació sigui automàtica (no podem generar una a una, i inclús, tampoc només d'un fitxer concret, la prova de perplexitat genera directoris dins de directoris).

Com que ja tenim tot el projecte en *python* i aquest ofereix moltes eines per parsejar fitxers al sistema, es pot fer un *script* que compleixi totes aquestes condicions.

El que s'ha fet, doncs, és un script que donat un fitxer *json*, et generi una imatge amb una matriu de similaritat i un mapa de calor, a més, aquest *script* recorre els directoris recursivament i va deixant el seu *output* la mateixa estructura de carpetes deixant les imatges on toca.

D'aquesta manera, comparar un gran nombre de matrius serà molt més ràpid.

10.2.3. Processament agregat de resultats

Hem volgut processar una gran quantitat de resultats tal que amb el mètode anterior seria inviable, per tal de poder treure conclusions a una escala major hem de poder agregar els resultats d'una manera que després puguem agrupar per mètodes.

Tal com hem comentat a una secció anterior (de com s'emmagatzemen els resultats) tenim per cada fitxer els diferents paràmetres, d'aquesta manera poder discriminar per tipus de paràmetre i veure els resultats agregats.

Aquests resultats es va optar per agafar fulls de *jupyter notebook* i executar per blocs, ja que molt més ràpid el desenvolupament aquí que anar tota l'estona executant el codi des del terminal.

11. Escalabilitat del projecte

Abans d'arribar a l'experimentació hem de veure com poder escalar el projecte per tal d'afegir nous elements en cas que es necessiti durant l'experimentació.

A aquesta secció s'explica com es poden fer iteracions sobre el projecte per tal de poder afegir conceptes o reajustar paràmetres. Això vol dir que si ens hem deixat paràmetres, contexts, models o mètriques podem arribar a iterar sobre el projecte per tal d'afegir noves característiques i estendre els resultats per obtenir noves conclusions.

Hi ha tres grans elements al projecte: Mètrica, Context i Model. Aquí es detalla com afegir nous elements dels dos primers tipus esmentats, no es detalla com afegir nous elements del tipus Model, ja que no entra a l'abast del projecte, però en algun moment si s'havia contemplat afegir un altre tipus de Model.

Finalment, també tenim un tipus d'iteració per tal de reajustar els paràmetres del model en cas que els resultats aconseguits no hagin sigut satisfactoris.

11.1. Cicle per reajustar paràmetres

Ha passat que en un principi els paràmetres no eren els correctes o que es volia provar amb altres paràmetres en llegir el model. No podrem reajustar el model de wikipedia, ja que és un model extern preentrenat, però si podem fer-ho amb el de blogs, científic o qualsevol que vulguem afegir mitjançant un cicle per introduir un nou context.

Aquest reajustament és independent de la pipeline d'execució del projecte, ja que una vegada entrenat el model amb els paràmetres que necessitem, en carregar-lo no hem d'especificar cap paràmetre, a l'estat del model al fitxer ja s'emmagatzema això i tot l'entrenament s'ha fet amb aquests paràmetres.

Haurem de decidir sobre quin model volem modificar els paràmetres primer.

11.1.1. Trobar uns paràmetres correctes

Els paràmetres a ajustar estan comentats a cada secció corresponent de cada model, en aquest cas diem trobar, ja que fixar uns paràmetres aleatoris no sempre és el més eficient, ja que hem de veure com de costós computacionalment és entrenar aquest model amb aquests paràmetres. Ens hem de fixar també com és de precís i això només es pot fer avaluant a posteriori el model.

11.1.2. Entrenament del model

Evidentment, hem d'entrenar el model una vegada fixem els paràmetres, el codi a ajustar està dins de la secció de cada paquet de model. Es dona una ruta que conté els documents a entrenar i un objectiu que és on es guardarà el model, el path aquest serà la referència als documents d'un context determinat, per exemple, si volem entrenar el model amb el context de blogs o el científic.

Per cada context que tinguem haurem d'entrenar els nous models per tal de donar homogeneïtat als resultats, això és molt important, ja que una de les premisses és que tots els contextos són entrenats sobre els mateixos paràmetres.

11.1.3. Executar el test de similaritat

Per tal d'assegurar que el model és efectiu tenim el test de similaritat (explicat a la secció de tests), aquest test ens assegura, com a mínim, que dos documents molt similars en fer una mesura de distància entre els *embeddings* d'aquests dos documents serà bastant petita i de dos documents molt diferents serà molt gran. Es detalla en profunditat a una de les seccions següents.

11.1.4. (No Implementat) Trobar un òptim local pels paràmetres

Aquesta secció no va donar temps a implementar-la, la idea era que donat un conjunt de paràmetres provar totes les combinacions, avaluar cadascuna d'aquestes i seleccionar el model que millor s'ha comportat. Per fer aquesta avaluació s'ha pensat en un mètode de predicció comentat a continuació.

11.1.4.1. Mètode de predicció per avaluar models

La idea és trobar una mètrica que ens digui com pot ser d'acurat aquest model, el mètode a seguir és agafar un conjunt de documents i separar-los per frases o quantitat de *tokens*, fet això, hem de generar forats de paraules que hauran de ser predits pels models, fet això haurem de fer un coeficient d'encerts i fallades per model, evidentment per cada model s'hauran d'agafar els mateixos documents.

Un exemple seria:

Tenim el següent document:

"a b c d e f g h i j k l m n o p q r s t u"

Amb una mida de forat 3 i una mida de frase 5, se'ns generaria el següent esquema:

(["a","b"], ["c","d","e"]),
(["f","g"], ["h","i","j"]),
(["k","l"], ["m","n","o"]),
(["p","q"], ["r","s","t"]),
(["u",""], ["","","",""])

La primera columna és el que farem servir per predir, la segona la resposta correcta de predicció.

Una mètrica ràpida és veure encerts de cada model i agafar el model amb més encerts.

Evidentment, estem limitats per les capacitats computacionals i la quantitat de textos que podem fer servir per fer aquesta avaluació, per això es diu que és un òptim local de paràmetres.

11.2. Cicle per introduir un nou context

Aquesta fase implica totes les components del projecte, introduir un context.

11.2.1. Trobar un dataset i preprocessar

S'ha de trobar un bon dataset que inclogui una bona quantitat de documents i si pot ser, la temàtica o tipus sigui homogeni en aquest. Si fem servir un context massa global el que trobarem és que les avaluacions que podem fer d'aquest respecte a altres contextos quedaran una mica diluïdes.

Posant un exemple dels contextos que fem servir, el de wikipedia ens serveix com un context molt general i poder fer avaluacions dins d'aquest per tal de fer avaluacions molt més generals, a una de les seccions següents (Comparació entre context general i context específic) tenim una bona comparació de com afecta entrenar un model general i entrenar un model específic.

Podem també fer feina comparativa de freqüències i veure histogrames, així també podem conèixer millor aquest dataset.

11.2.2. Entrenament dels models

Haurem de crear una nova referència pel nou model i entrenar un model per *Word2Vec* i *Doc2Vec*, haurem també d'executar el test de validació.

Una vegada fet tot això podem fer comparacions amb documents d'altres contextos en aquest model i veure com es comporten. Idealment, és que a l'hora de veure el *scatterplot* estiguin agrupats els diferents documents del mateix tipus, si no és així, probablement col·lapsaran temàtiques.

11.2.3. Actualitzar referències a la pipeline d'execució

Tenim diversos punts on s'especifica o fa servir coses dels documents.

1) Referència de l'enumeració Context que és la que es fa servir a l'hora d'indicar una prova.

2) Càrrega de documents per la prova: fins ara teníem dos tipus de documents, de blogs i científics, a l'hora d'executar un test sempre agafàvem meitat i meitat de cada, ara l'escenari ha canviat i, per tant, haurem d'afegir dos paràmetres a l'execució d'aquestes proves per indicar quins dos tipus de documents volem fer servir per fer aquesta avaluació, per tant, podrem escollir un context que no tingui res a veure i dos tipus de documents diferents i fer la prova.

Només fem servir dos tipus de documents a l'hora de fer les proves, encara que podríem fer servir més i fer-ho per segments. Un altre possibilitat és donar un *array* de tipus de documents i el nombre total de documents quedaria dividit entre el nombre de tipus de documents passats per paràmetre tenint segments, el que per aquesta funcionalitat hauríem d'adaptar la prova de mitjanes per quadrants.

3) Càrrega del model, haurem de crear una altra referència pel nou model a carregar en *Doc2Vec* i *Word2Vec*.

Fet això ja podem fer servir el nou context creat.

11.3. Cicle per introduir una nova mètrica

Aquest cicle és molt més simple, ja que només hem d'actualitzar la referència a l'enumeració de *Metric* i afegir el nou mètode que, des de dues llistes de *floats* (dos punts en l'espai euclidià, l'índex a cada llista indica el punt) ha de calcular la distància entre ells.

No requereix res més i ja es contempla a tota la resta de l'*output* que pot existir una determinada mètrica.

12. Experimentació

Finalment, hem arribat a l'experimentació, al llarg de la memòria hem fet un repàs dels conceptes bàsics per tal d'entendre diversos conceptes que s'han emprat i s'empraran a aquesta secció, hem vist totes les mètriques, contexts i models que farem servir als experiments, també hem explicat la *pipeline* de preprocessament de text, com entrenar els dos models principals del projecte i per últim com fer servir la rutina principal per a executar els experiments.

A aquesta secció veurem els diferents experiments que s'han realitzat:

- **Validació del model**, que és bàsic per tal de verificar que el model es comporta correctament.
- **Proves ràpides visuals**, diversos núvols de punts per veure com s'estructuren els diferents models entrenats i fer algunes comparacions d'*embeddings* reduint dimensionalitat.
- **Proves avançades fetes**, es comencen a comparar documents mitjanant *embeddings* i matrius de similaritat, aïllats i agrupats per classes.
- **Estructura de proves combinatòries**, execucions de la rutina principal per tal de generar una gran quantitat de comparacions agrupades, transformar els resultats a mapes de calor i comparar en taules els resultats.
- **Prova de perplexitat**, per tal d'assegurar les mateixes condicions amb totes les mètriques a les execucions massives s'ha inclòs un pas previ avaluació per tal de buscar un paràmetre adient de perplexitat.
- **Proves finals amb perplexitat 4**, experiment final amb la perplexitat trobada a la prova de perplexitat, s'executen comparacions massives i s'avaluen per grups discriminats amb *scripts* de *Python*.

12.1. Validació del model

Amb *Word2Vec* i amb *Doc2Vec* s'ha fet un *script* que, entre dos documents, on s'agafen (es conserven a l'original) i s'afegeixen a l'altre, amb això s'ha d'obtenir una similaritat creixent, això s'ha fet fent servir la mesura de distàncies *cosine similarity*. L'exportació s'ha fet en *CSV*.

La idea d'aquesta validació és veure si el model està correctament entrenat, molt al principi del projecte quan no es sabia fer servir el model era molt útil per fer prova i error. A les alçades que estem, és un "estàndard de qualitat" que ha de passar tot model que s'hagi de fer servir.

El que es fa exactament és, llegir dos fitxers, carreguem el model, aplica la *pipeline* de preprocessament, i fa 20 iteracions, a cada iteració el que es fa és, calcular la similaritat actual i copia 500 paraules d'un document a l'altre. A cada iteració el condicionant és que la similaritat ha d'incrementar cada vegada més.

Amb aquesta validació tenim un primer test de què com a mínim el model té un comportament mínim.

12.2. Proves ràpides visuals

A aquesta secció es detallen diversos tipus d'experiments per tal de veure com s'han comportat els diferents models a l'espai euclidià que s'ha format derivat dels entrenaments dels models.

12.2.1. Visualització tsne de tots els tokens del model

Aquesta prova consisteix a agafar els **x** primers *tokens* del vocabulari del model, per cada *token* agafat, obtenir el seu *embedding*, com que aquest *embedding* és **N** dimensional llavors aplicar-li un *TSNE* a tots aquests *embeddings*, finalment exportar en *CSV* i visualitzar al *D3 scatterplot*.

Les característiques en comú de totes aquestes visualitzacions són:

- Els *tokens* als quals no es troba molta relació amb altres queden a la perifèria del gràfic.
- Els *tokens* d'un mateix idioma queden en una agrupació més o menys gran i dins d'aquesta formen agrupacions més petites depenent les relacions que s'hagin trobat.
- Que dos *tokens* estiguin a prop no necessàriament han d'estar relacionats a la vida real, o que inclús que s'hagi trobat una relació entre aquests dos, en moltes ocasions, simplement relacionant un token amb altres, l'assignació a l'espai que se li ha donat ha pogut ser similar a la d'altre *token* que no tingui res a veure.
- Evidentment, les millors relacions són les que tenen tokens superposats, en aquest *plot* no es poden veure clarament, ja que es barregen les lletres a l'hora de visualitzar-ho (no es va perfeccionar perquè no es va considerar necessari invertir temps a crear un *plot* que faci alguna mena d'animació que escampi els textos en passar el ratolí).

12.2.1.1. Word2Vec

12.2.1.1.1. Blogs

En aquesta visualització podem veure com els *tokens* s'agrupen sobretot en grups circulars i de bastant densitat, i sobretot, els tokens no estan del tot agrupats per temàtiques si no por sembla ser, temes de conversa, això es pot inferir del tipus de *dataset*, al ser un *dataset* de posts i blogs a internet dels 2000's els temes de conversa involucren múltiples termes de tot tipus de idees, a més es podria interpretar que cada grup de *tokens* més o menys gran és una línea de pensament concreta (no necessàriament política) que comparteixen diversos usuaris que hagin escrit en aquests fòrums.

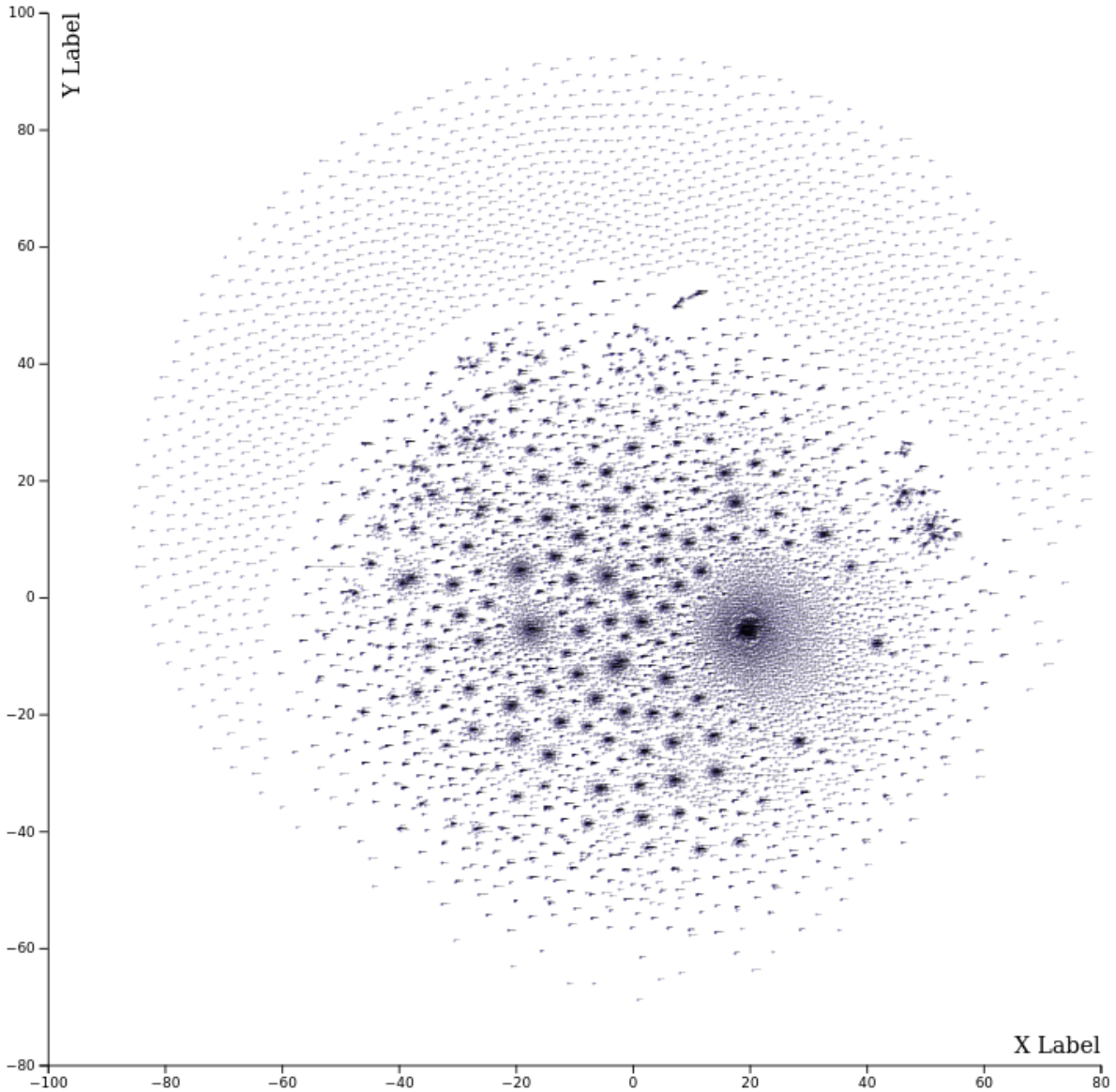

Figura 43: núvol de punts amb tsne de tots els tokens del model Word2Vec-Blogs.

Queda veure exemples que reforcin aquesta teoria, hem pres unes mostres aleatòries de grups de punts:

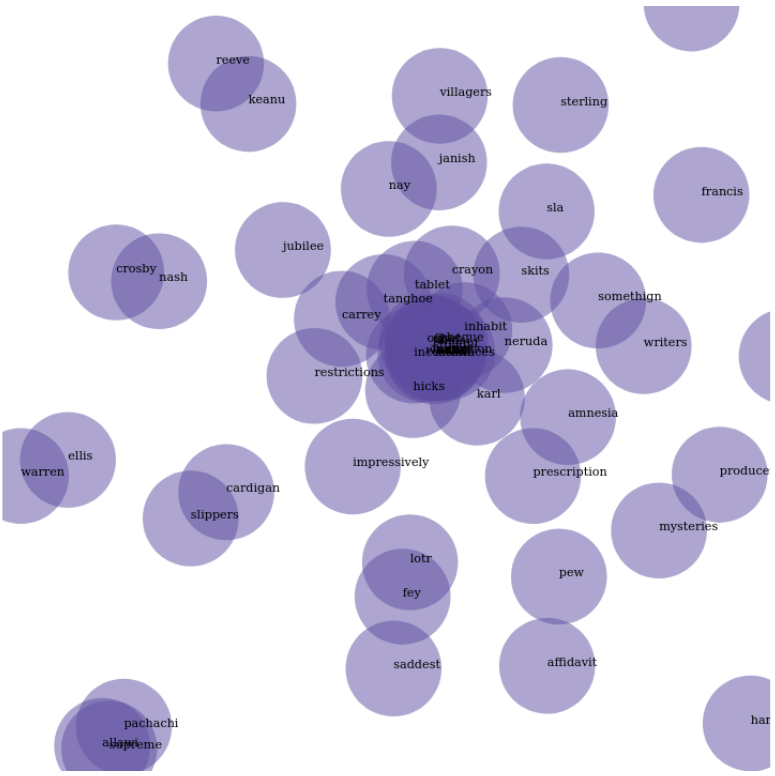

Figura 44: ampliació 1 núvol de punts amb tsne de tots els tokens del model Word2Vec-Blogs.

Topic de *matrix*, en aquella època va sortir la pel·lícula i la gent sembla que parlava molt d'ella.

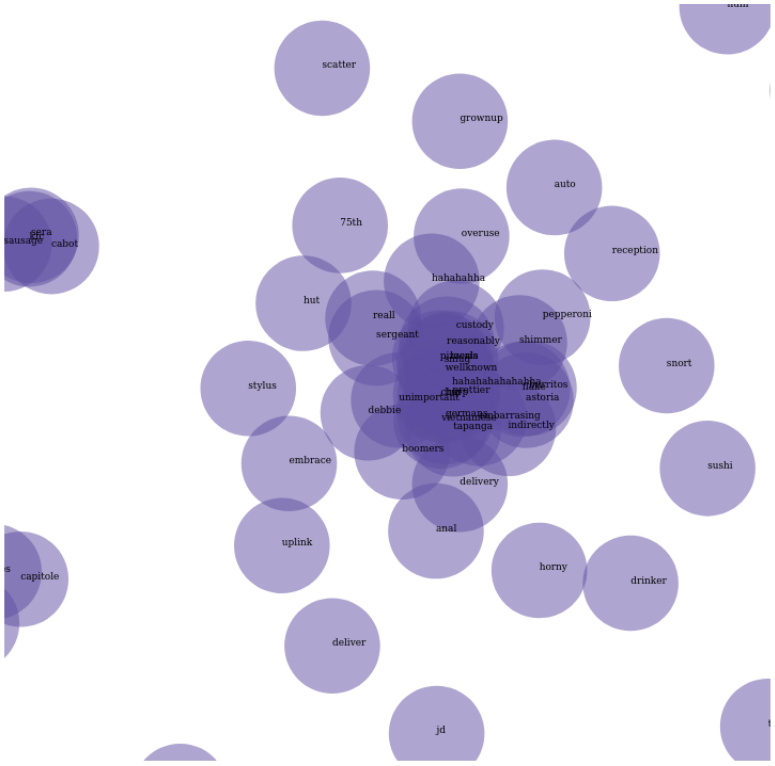

Figura 45: ampliació 2 núvol de punts amb tsne de tots els tokens del model Word2Vec-Blogs.

En aquest no sabria distingir molt bé la temàtica, potser té a veure amb cites entre persones.

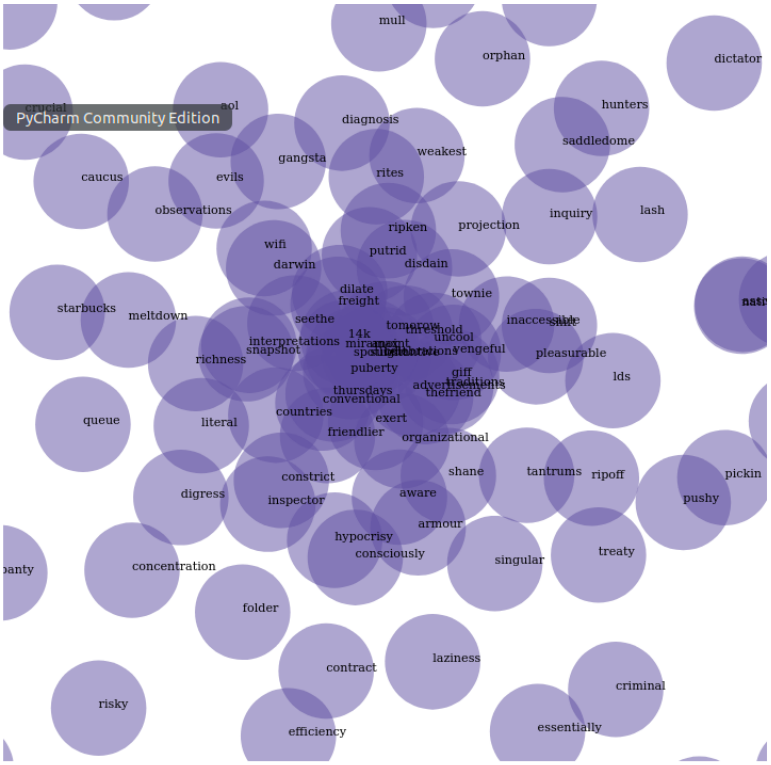

Figura 46: ampliació 3 núvol de punts amb tsne de tots els tokens del model Word2Vec-Blogs.

Sembla el llenguatge parlat d'un grup d'amics per quedar en algun lloc.

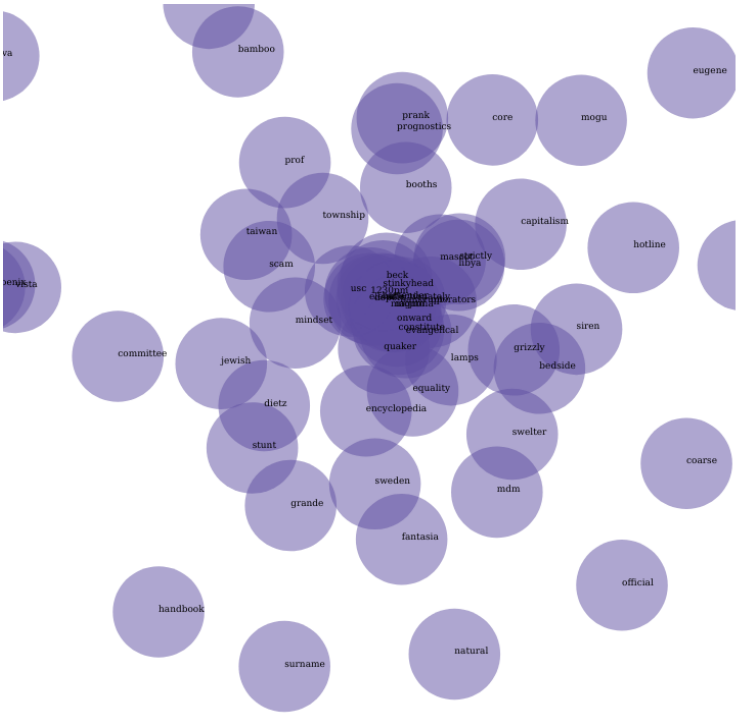

Figura 47: ampliació 4 núvol de punts amb tsne de tots els tokens del model Word2Vec-Blogs.

Sembla un tema sobre política i economia.

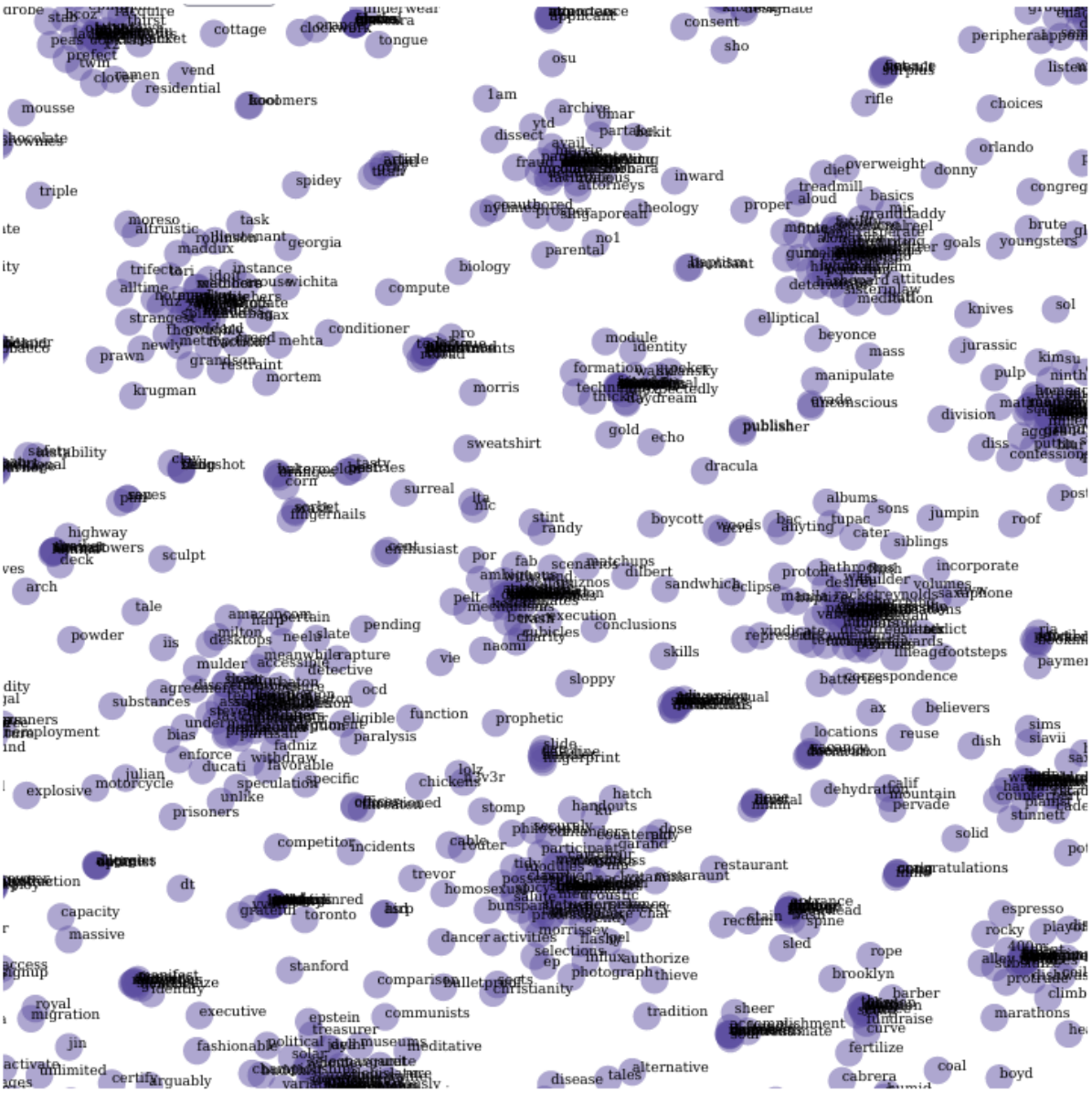

Figura 48: ampliació 5 núvol de punts amb tsne de tots els tokens del model Word2Vec-Blogs.

A aquesta imatge he volgut ficar una vista una mica més allunyada per veure una mica com es relacionen els diferents *topics* i les paraules una mica més aïllades.

12.2.1.1.2. Científic

En aquest context tenim agrupacions més petites i uniformes, pot ser degut al fet que cada document pot fer servir termes molt específics i entre documents hi ha temàtiques molt dispars, per exemple, un document pot parlar de com es distribueix la renda per barris i un altre de les òrbites dels planetes al sistema solar, llavors és normal veure aquesta distribució.

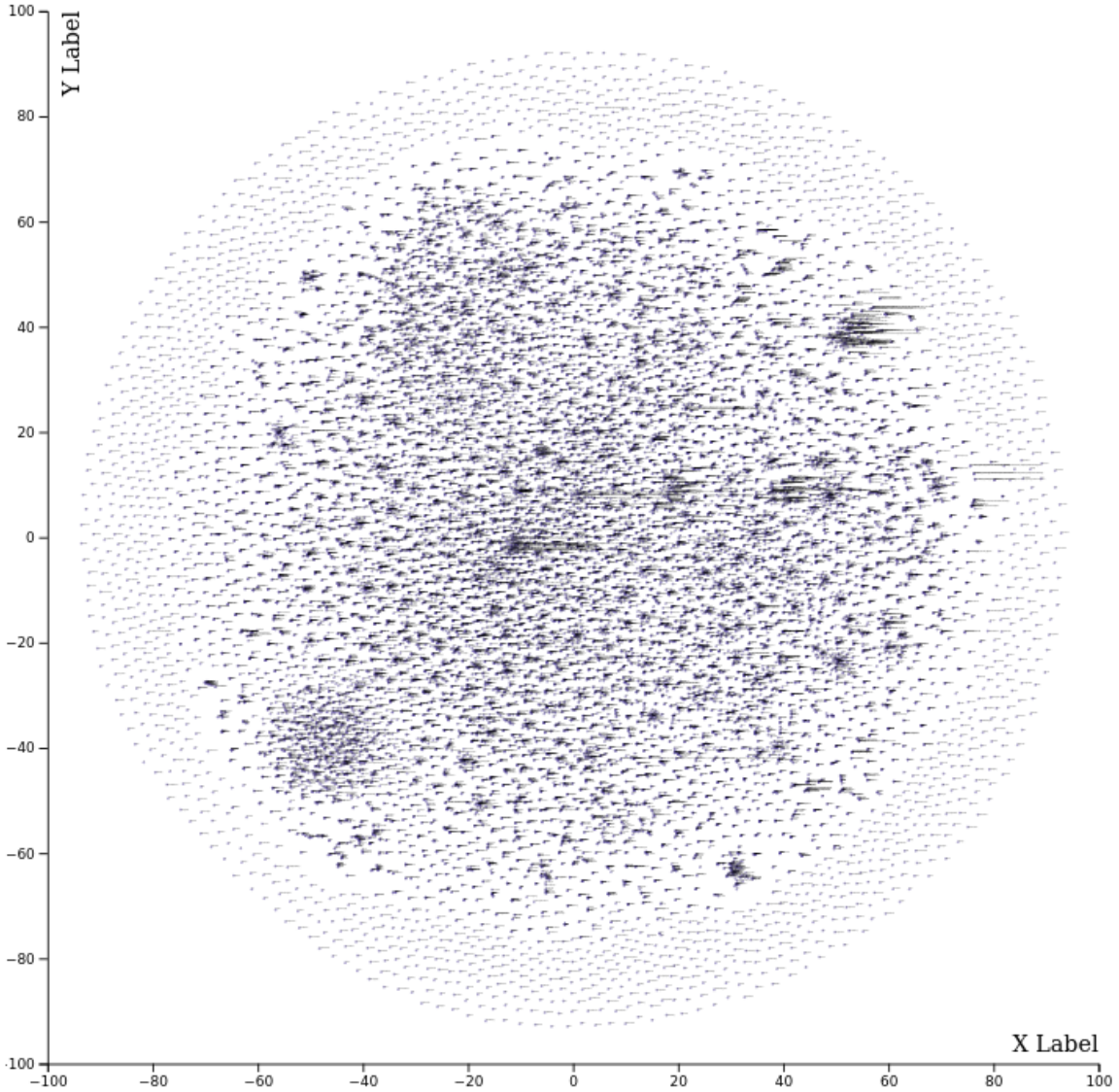

Figura 49: núvol de punts amb tsne de tots els tokens del model Word2Vec-Scientific.

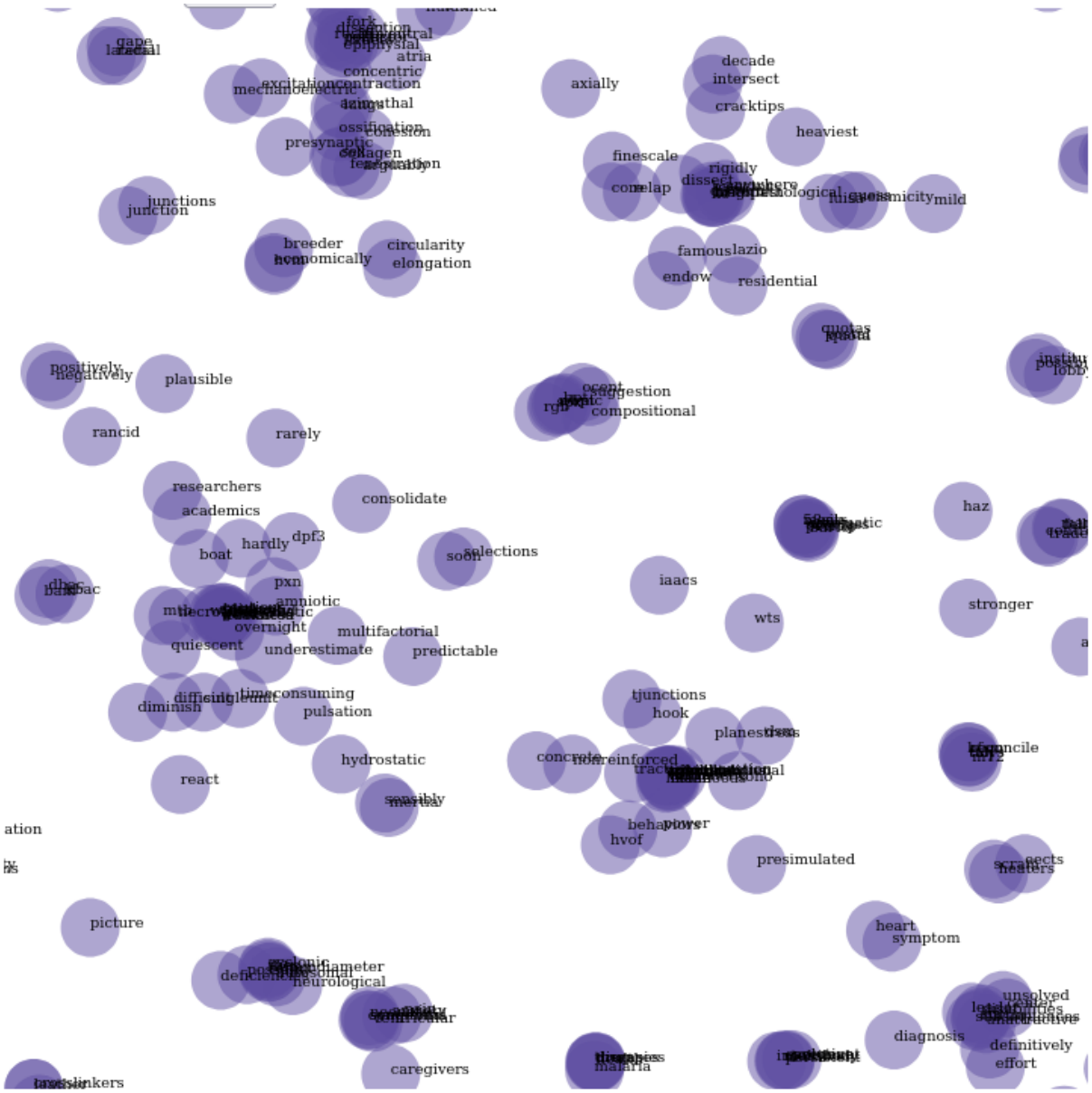

Figura 50: ampliació núvol de punts amb tsne de tots els tokens del model Word2Vec-Scientific.

12.2.1.1.3. Wikipedia

En aquest model s'han limitat el nombre *tokens,* ja que en fer una visualització per *tsne* el temps de còmput era molt elevat, la visualització és d'una mostra i, per tant no es pot avaluar correctament mitjançant aquest mètode, però si podem fer una estimació visual de com s'ha comportat, els grups que veiem són bastant més diversos i diferenciats, sobretot no hi ha tantes paraules sense grup, podem inferir d'aquí que a l'haver-hi molta més informació a l'hora de fer l'entrenament és més probable que per qualsevol paraula es trobi una relació amb altre, wikipedia conté tota mena de documents, per tant, és normal trobar grups de diverses mides i tipus (per exemple, hi han grups que són més concentrats i altres que són més circulars amb més dispersió).

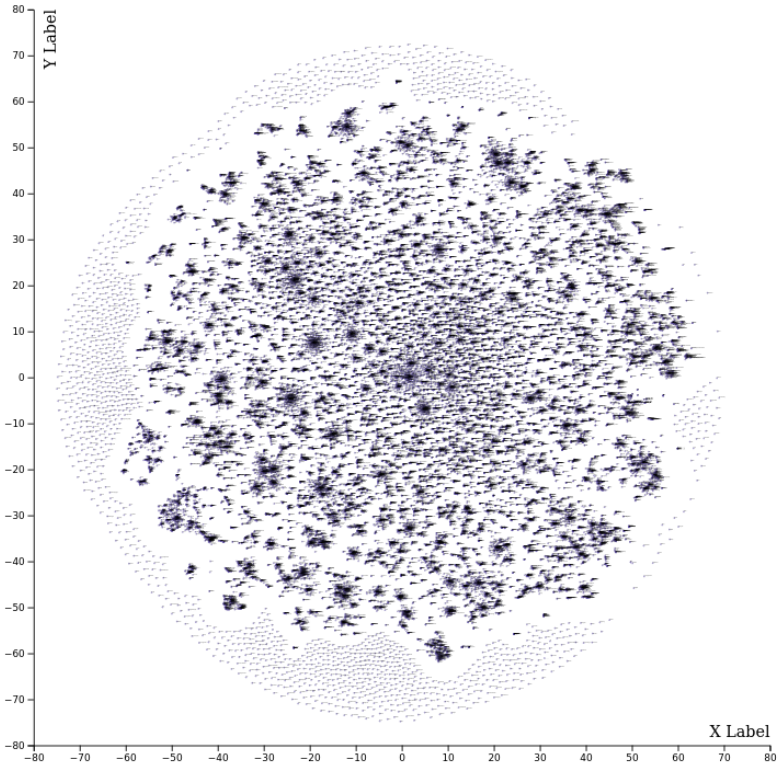

Figura 51: núvol de punts amb tsne de tots els tokens del model Word2Vec-Wikipedia.

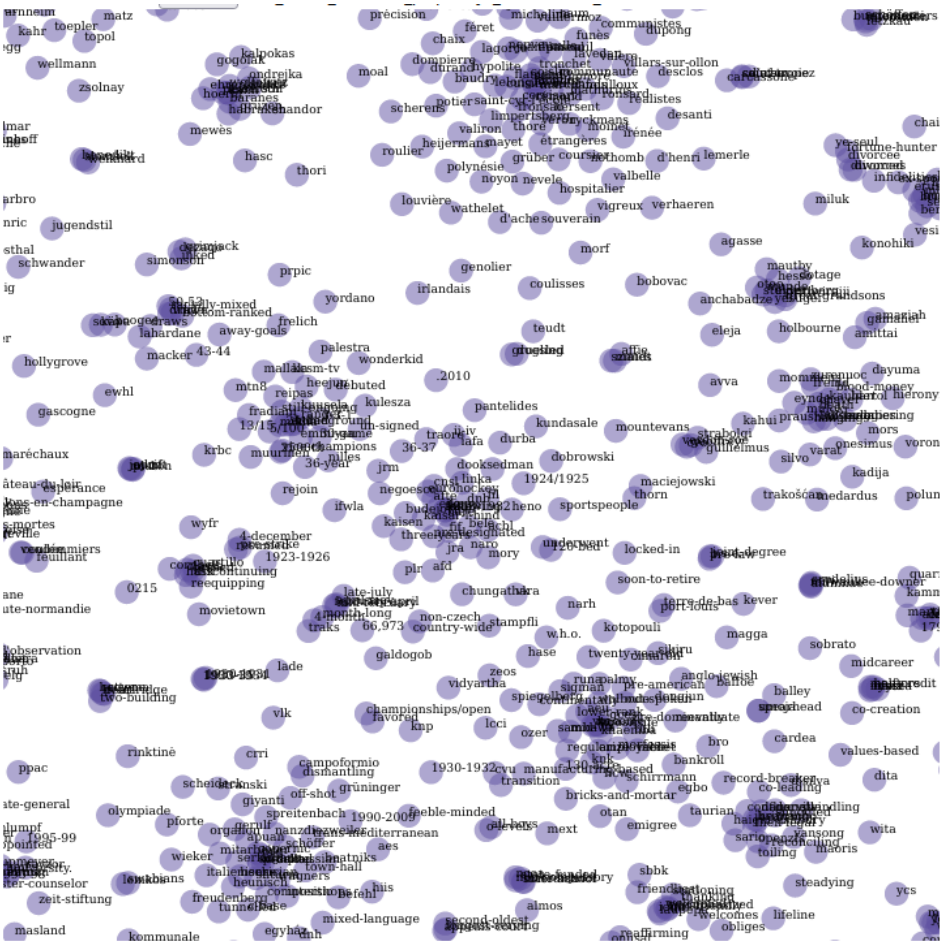

Figura 52: ampliació 1 núvol de punts amb tsne de tots els tokens del model Word2Vec-Wikipedia.

12.2.1.2. Doc2Vec

A *Doc2Vec* observem un comportament similar al de *Word2Vec*, no hi ha molt a recalcar.

12.2.1.2.1. Blogs

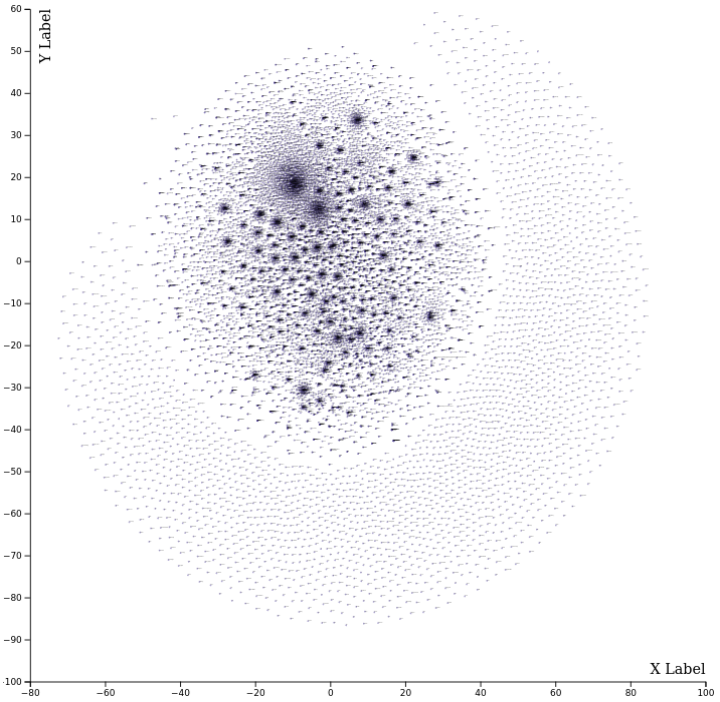

Figura 53: núvol de punts amb tsne de tots els tokens del model Doc2Vec-Blogs.

12.2.1.2.2. Scientific

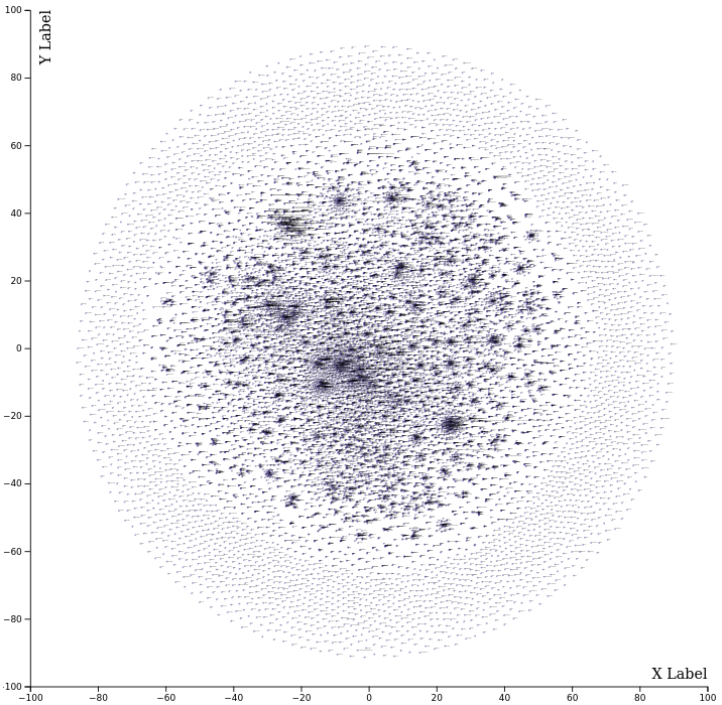

Figura 54: núvol de punts amb tsne de tots els tokens del model Doc2Vec-Scientific.

12.2.1.2.3. Wikipedia

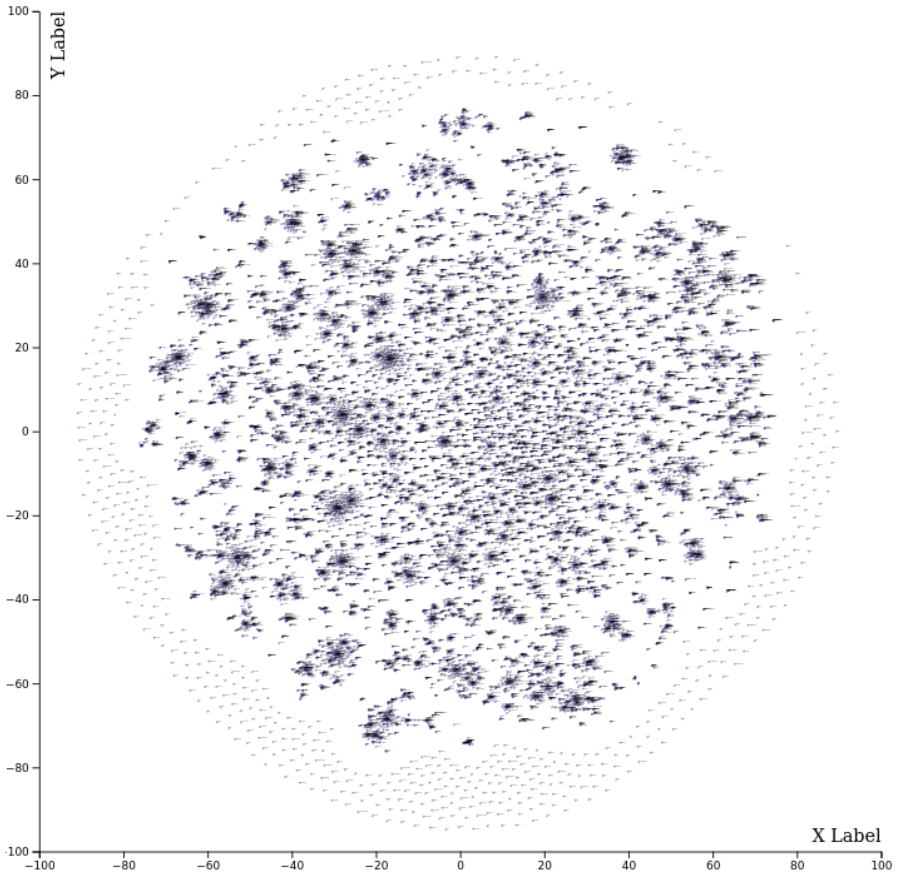

Figura 55: núvol de punts amb tsne de tots els tokens del model Doc2Vec-Wikipedia.

12.2.2. Visualització dels tokens de n documents per classes (classe per document)

Aquesta visualització agafa els **n** primers documents de blogs i els **n** primers documents científics, per cada document es filtren els *tokens* que no hi siguin al vocabulari (d'altra manera seria impossible calcular-ho) i per cada document i *token* agafa l'*embedding* del *token* (només del *token*, no del document) i li assigna una classe. Una vegada té tots els *embeddings* li aplica *TSNE* i exporta a *CSV*. Les classes assignades són per document, els punts de color vermell (per exemple) pertanyen al mateix document.

En aquest assaig es visualitzen els primers 5 documents de blogs i els primers 5 documents científics, veiem com no es diferencien massa al context de Blogs amb perplexitat 4.

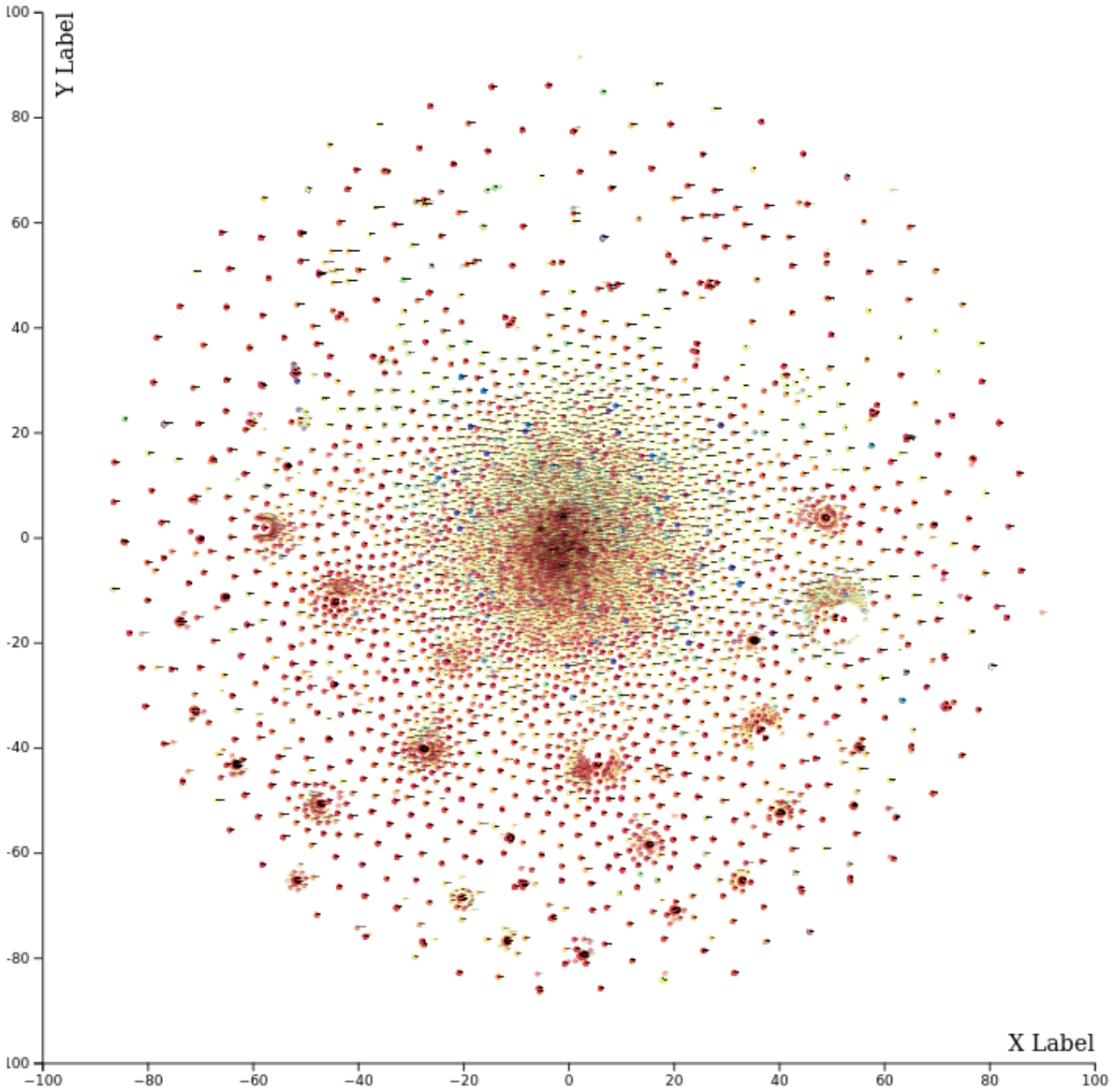

Figura 56: núvol de punts amb tsne perplexitat 4 dels tokens de n documents per classes Word2Vec-Blogs.

Provem al context Wikipedia amb perplexitat 4 que és més general:

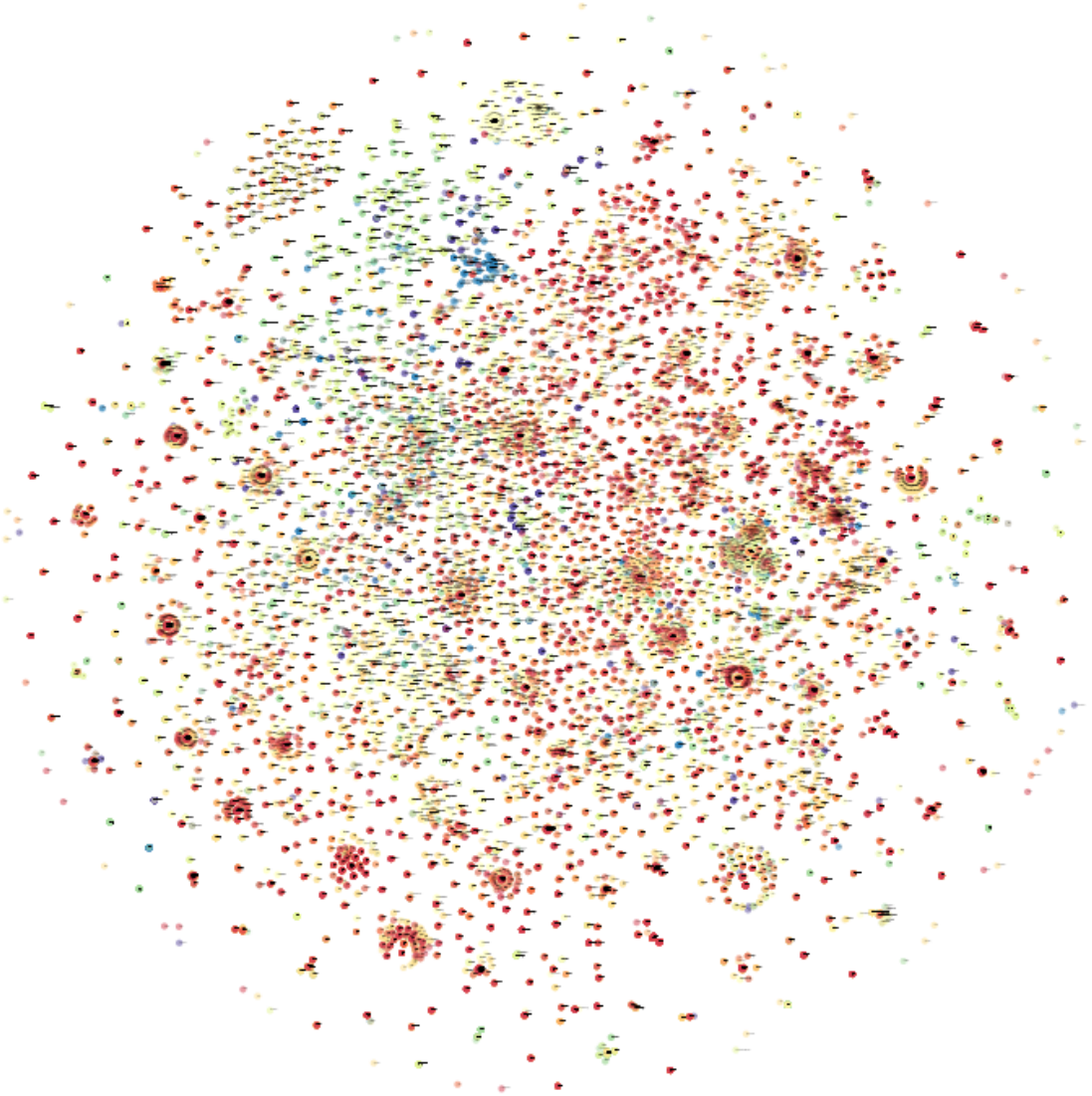

Figura 57: núvol de punts amb tsne perplexitat 4 dels tokens de n documents per classes Word2Vec-Wikipedia.

Veiem que s'agrupen una mica diferent que a l'anterior, ara agafem dues mostres de dues classes diferents:

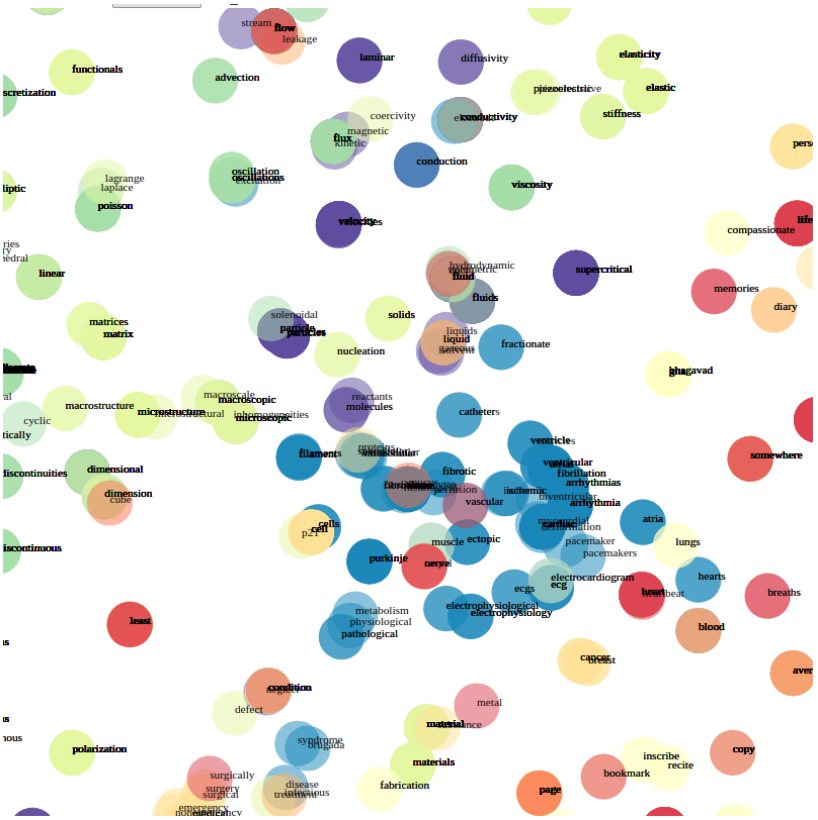

Figura 58: ampliació 1 núvol de punts amb tsne perplexitat 4 dels tokens de n documents per classes Word2Vec-Wikipedia.

En aquesta mostra podem veure la classe blava que evidentment parla d'alguna cosa relacionada amb la medicina, les paraules són agrupades bastant properes entre elles, això és degut al fet que els altres documents no inclouen *tokens* d'aquest tipus, però veiem que realment és indiferent la classe, en aquesta prova importa la mateixa paraula i on es posiciona, que hàgim trobat un grup aïllat no vol dir res més que això.

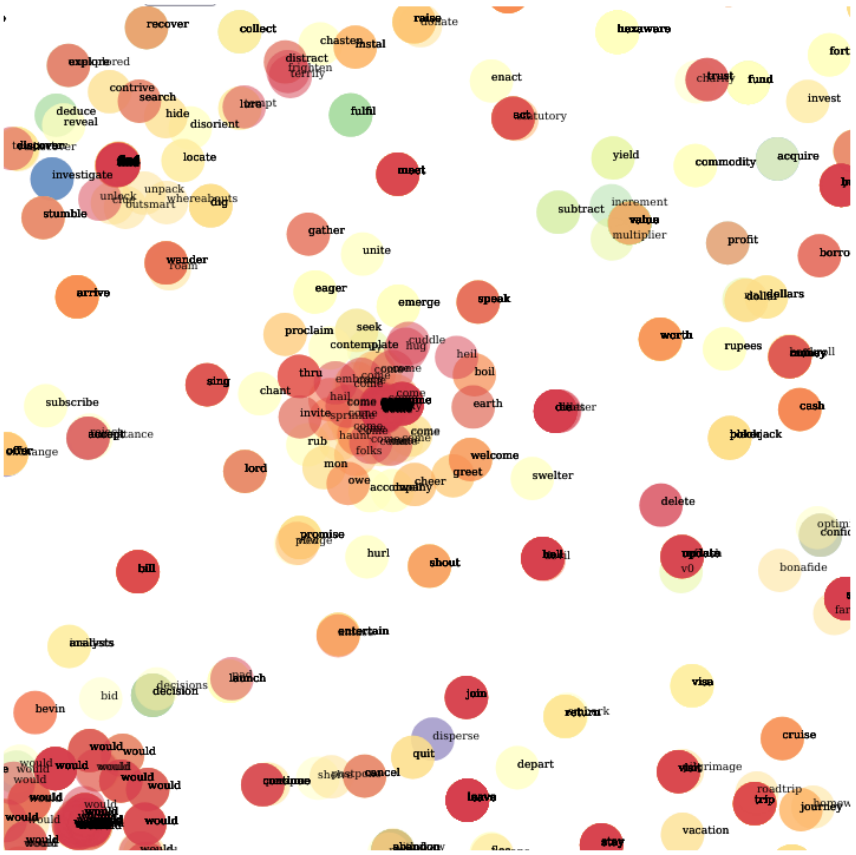

Figura 59: ampliació 2 núvol de punts amb tsne perplexitat 4 dels tokens de n documents per classes Word2Vec-Wikipedia.

A baix a l'esquerra, podem veure la mateixa paraula moltes vegades repetida a un grup, això és degut al fet que *tsne* agrupa segons les dades que té, quan extraiem la paraula del model ens dóna les mateixes coordenades ho fem **1** o **n** vegades, però segons com funciona *tsne* a l'hora de reduir dimensionalitat no conserva la posició original i per això veiem diferents posicions de la mateixa paraula en comptes d'estar compactes a un punt.

12.2.3. Visualització d'embeddings de n documents per classes (classe per tipus de document)

En aquesta visualització el que es fa és agafar els **n** primers documents per classes, o sigui, classe 1 per documents de blogs i classe 2 per documents científics. Es calcula *l'embedding* de cada document i es fa un *TSNE* per reduir la dimensionalitat, finalment s'exporta a *CSV*.

Aquí hi ha només dues classes, o blog o científic, s'ha aplicat al context de Wikipedia que és el més general i perplexitat 4, veiem que es formen clarament dos grups de punts.

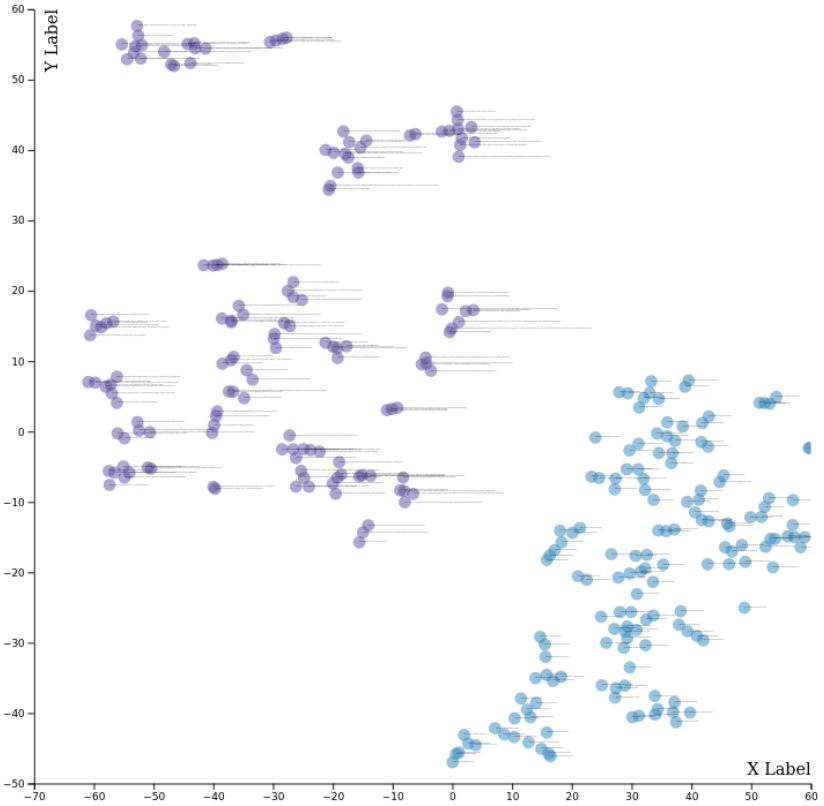

Figura 60: visualització d'embeddings de n documents per classes Word2Vec-Wikipedia.

Poc a ressaltar, veiem que fa la diferenciació correctament, ens pot servir per veure com estan distribuïts els punts quan fem una matriu de similaritat.

12.2.4. Visualització de top words mitjançant tfidf calculant embeddings token a token i fent la mitja per document classificat per classes (classe per tipus de document)

En aquesta visualització el procediment és el següent:

- Obté el *tfidf* per grups (classes), consultar l'apartat corresponent a l'algorisme al document.
- Filtra les paraules de cada document per tal que siguin al vocabulari del model.
- Aconsegueix els *embeddings* de cada paraula.
- Fa la mitjana de tots els *tokens* per document.
- Fa el *TSNE*.
- Exporta a *CSV*.

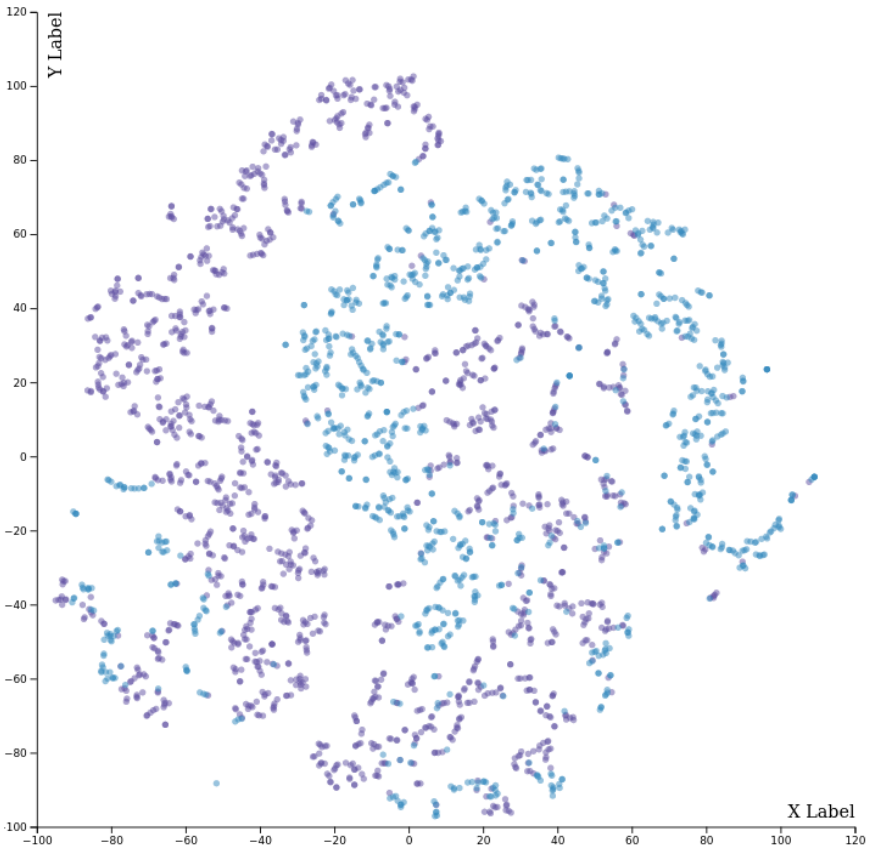

Figura 61: visualització de top words mitjançant tfidf calculant embeddings token a token i mitja classificant per classes amb tsne perplexitat 4 amb Doc2Vec-Wikipedia.

Generat amb *Doc2Vec*, 1500 documents, 100 *top words*, perplexitat 4 al context de Wikipedia (aquesta prova només està implementada amb *Doc2Vec*).

No ens diu molt aquesta figura, sí que veiem una clara diferenciació entre tipus de documents, però a la pràctica no és un mètode que ens serveixi de molt. L'objectiu d'aquest test era veure com es comportaven les dades amb una prova una mica més complexa.

12.2.5. Comparació entre context específic i context general

Volem veure com afecta un context general i un context específic, en aquesta secció veurem la comparativa entre aquests.

12.2.5.1. Context específic (Científic)

Veure si un model és efectiu amb dos *datasets* completament diferents, es podrà veure que els documents comparats entre les dues classes són molt distants, i els documents que siguin del mateix *dataset* seran molt propers. Si fem el mateix amb un context que tingui a veure amb un dels conjunts dels documents es veurà clarament molt diluït els resultats. Per exemple:

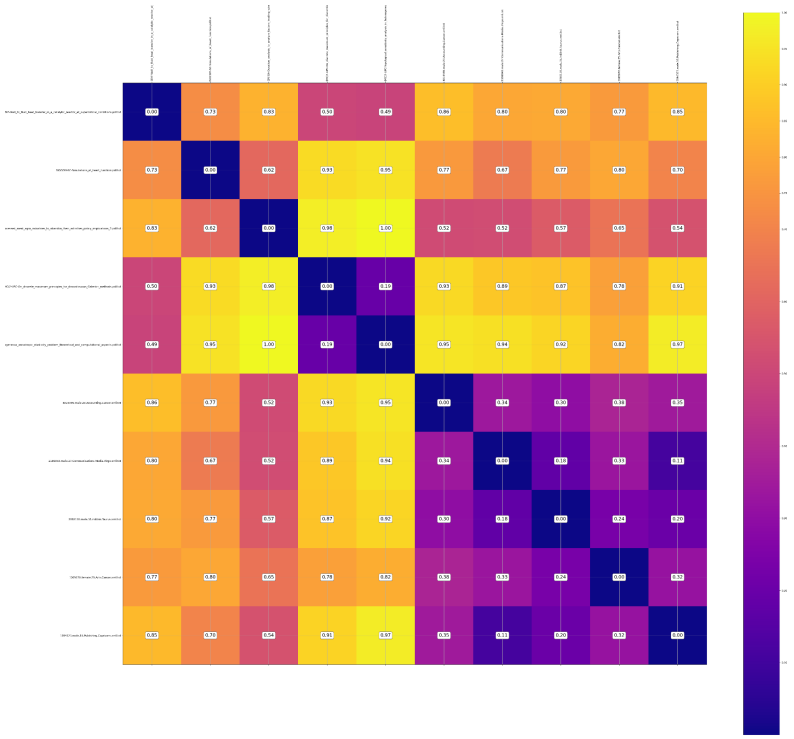

Figura 62: matriu similaritat Word2Vec-Científic mètrica Euclidiana.

Nom del fitxer: matrix_word2vec_scientific_euclidean_docs10__2021-04-23-15-10-02.json.png

El primer quadrant correspon a documents científics comparats entre ells, veiem com tenim disparitat de resultats, ja que el context engloba aquesta temàtica, en canvi, els documents de blogs estan com molt a prop. El lila indica que són a prop i el groc que són molt llunyans. Per veure il·lustrat aquest exemple hem generat un *scatterplot* amb les ubicacions d'aquests documents i uns quants més:

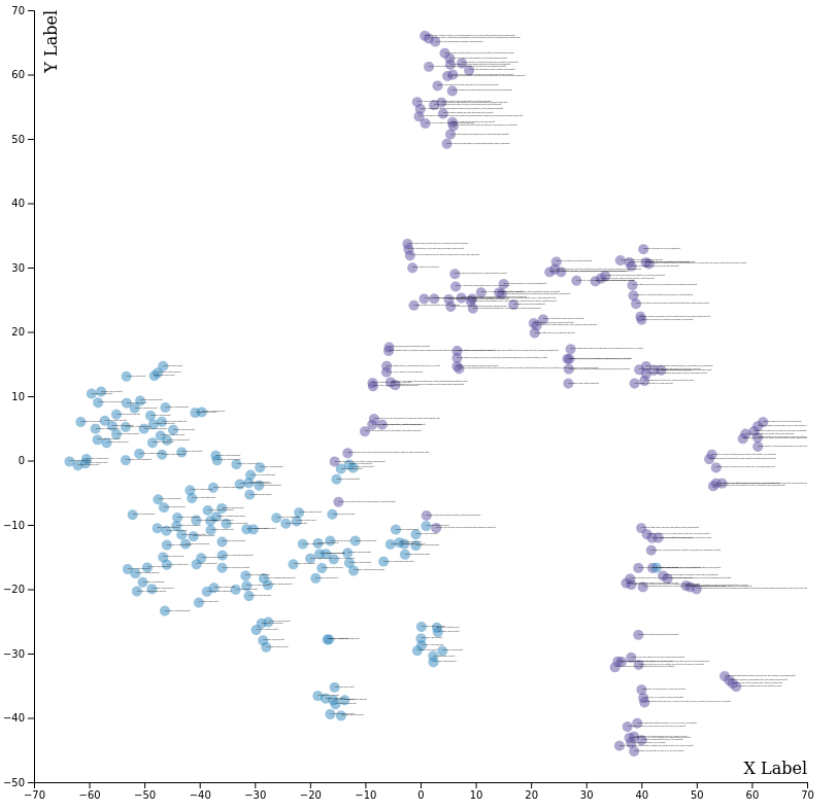

Figura 63: núvol de punts dels embeddings de n documents aplicat tsne perplexitat 4 amb Word2Vec-Científic per classes.

Aquest plot ens mostra la ubicació de cada *embedding* generat per document amb una reducció de dimensionalitat amb *tsne* i perplexitat 4, els punts blaus són documents de blogs i els punts liles documents científics. Podem veure clarament que la distància mitjana entre els documents científics és més elevada que la dels documents blogs. Evidentment, no coincideix amb la matriu de similaritat les distàncies perquè en fer la reducció de dimensionalitat amb *tsne* es perd la posició original.

12.2.5.2. Context general (Wikipedia)

Vist com es comporta en un model específic, passem a veure com es comporten les comparacions entre documents dins d'un model molt més general, anem a veure la matriu de similaritat:

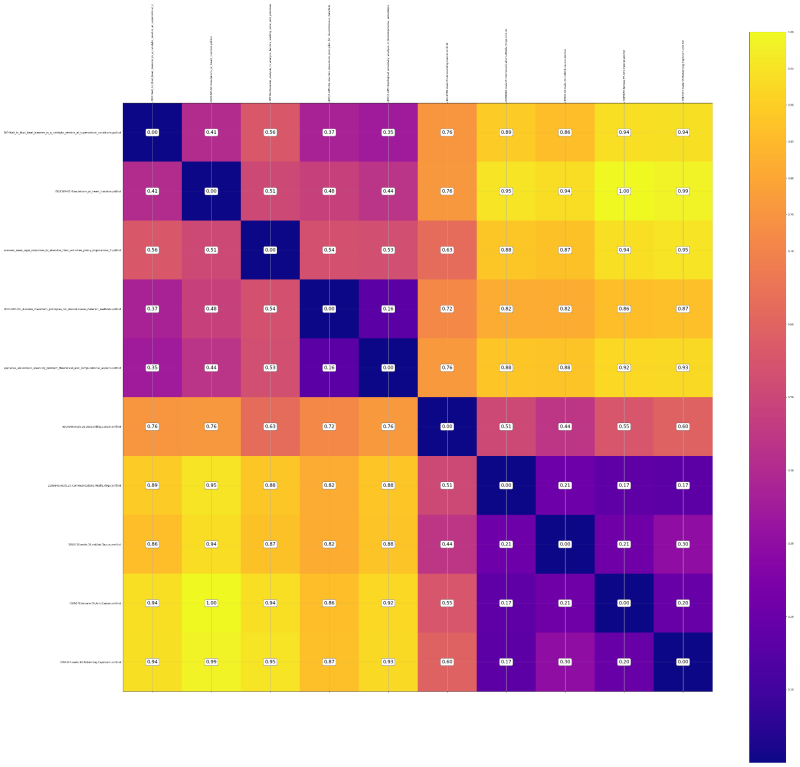

Figura 64: matriu similaritat Word2Vec-Wikipedia mètrica Euclidiana.

Nom del fitxer: matrix_word2vec_wiki_euclidean_docs10__2021-04-25-01-53-45.json.png

Veiem que les comparacions entre documents del mateix context són bastant més versemblants pels dos tipus de documents i les comparacions entre documents de diferents tipus bastant diferents.

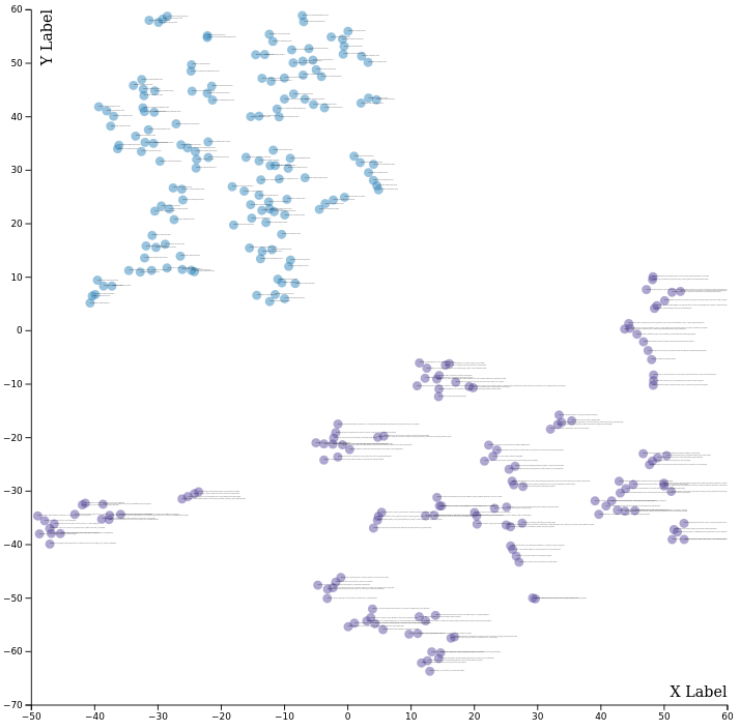

Figura 65: núvol de punts dels embeddings de n documents aplicat tsne perplexitat 4 amb Word2Vec-Wikipedia per classes.

Podem veure que les distribucions de diferents classes de punts són una mica més versemblants, no tenim en aquesta mostra tampoc punts d'entre classes barrejats o molt a prop.

12.3. Proves avançades

A aquests tipus de proves fem servir les matrius de similaritat per documents aïllats i classificats per classes, la idea és donar una noció de com es comparen els documents.

Comencem amb un primer experiment amb cosine similarity amb *Word2Vec* i els dos contexts Blogs i Científic; després veurem un exemple del *Word Mover Distance* i conclourem que no és viable fer-ho servir al projecte.

Finalment, acabem la secció veient com es fan les mitjanes de distàncies entre documents del mateix context per tal de poder agregar els resultats de matrius **NxN** molt grans i poder comparar similaritats entre classes de documents.

12.3.1. Distància entre documents mitjançant cosine similarity

Agafar **n** documents i fer similaritat entre ells, ens resulta en **n^2** comparacions i fer una matriu de similaritat. Després carregar documents, per cada document, obtenir *embedding* de cada document i per cada un fer comparació amb tots els altres.

La idea és agafar meitat del *dataset* científic i meitat del dataset de blogs, i fer la prova per els 3 contextos.

Bé, aquesta és una de les proves que s'executen dins del paquet de l'estructura de proves combinatòria, es deixen unes mostres en aquest apartat, però per veure un resultat complet s'ha de consultar l'apartat corresponent.

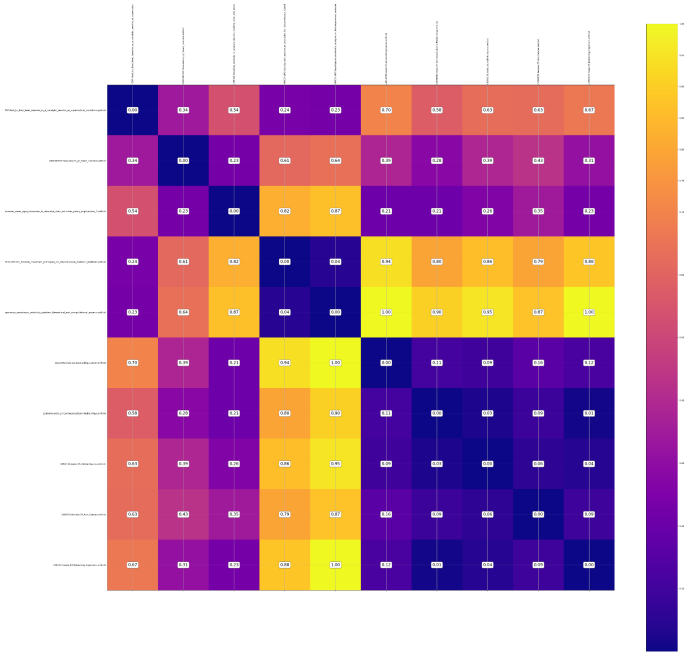

Figura 66: matriu similaritat 10 documents Word2Vec-Científic mètrica Cosine similarity.

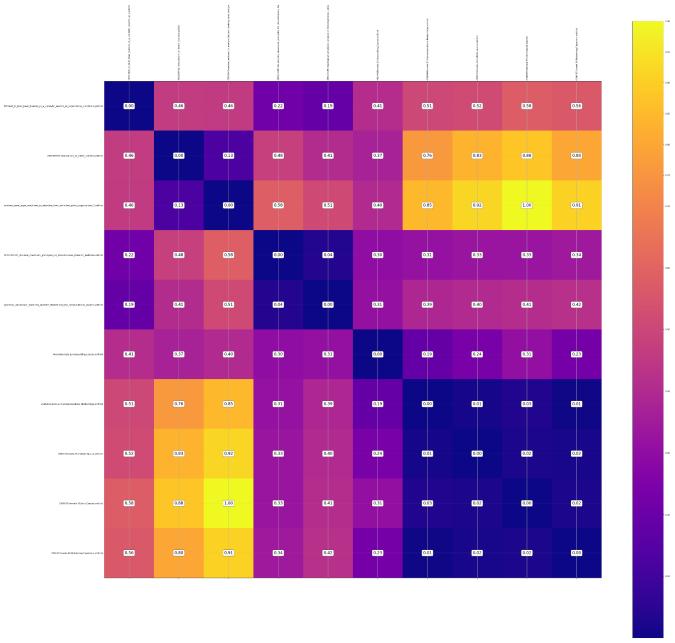

Figura 67: matriu similaritat 10 documents Word2Vec-Blogs mètrica Cosine similarity.

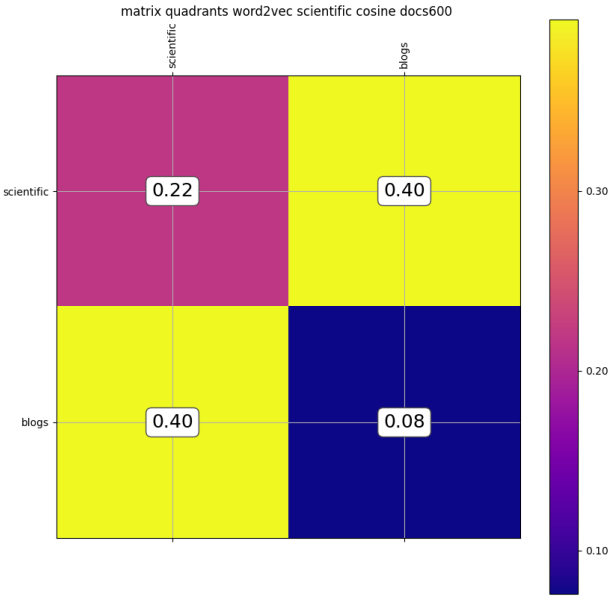

Figura 68: matriu similaritat per classes (300 documents) Word2Vec-Científic mètrica Cosine similarity.

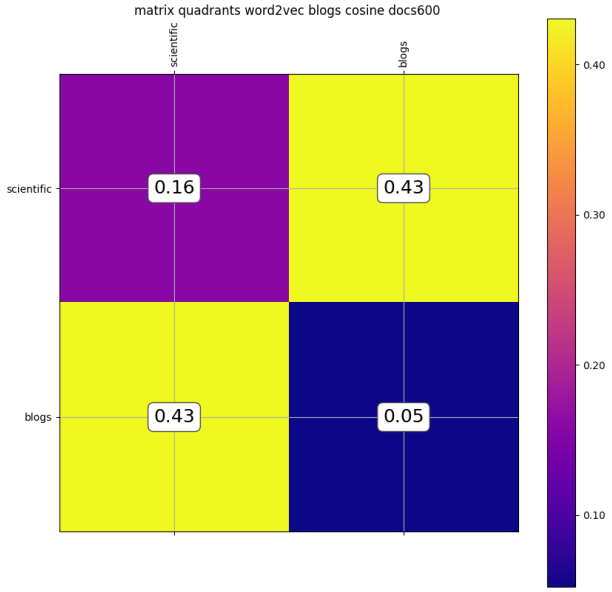

Figura 69: matriu similaritat per classes (300 documents) Word2Vec-Blogs mètrica Cosine similarity.

Podem veure que a les dues mostres inicials en el context científic es noten unes diferències més sòlides entre documents, però en fer la comparació per quadrants i la mitjana de les comparacions veiem que el context de Blogs és una mica més precís que el científic.

12.3.2. Distància entre documents mitjançant mètrica euclidiana

Mateix procediment que amb el *cosine similarity* però fent servir la distància euclidiana. A l'estructura de proves combinatòries es pot veure el resultat agregat (a la següent secció: Estructura de proves combinatòries).

12.3.3. Distància entre documents mitjançant Word Mover Distance

Aquesta mètrica és bastant més complexa, ja que ja no es fa directament amb l'*embedding* del document, sinó, que s'ha de fer amb els vectors de cada paraula representats al model en qüestió, amb això fer el càlcul, el que passa és que és un algoritme bastant lent, per tant, hem de fer una mètrica per classificar les paraules més representatives de cada document, la millor manera que tenim a mà és fer un *tfidf* i fer una fita per les paraules més rellevants. Llavors podem calcular en molt menys temps el *Word Mover Distance*.

En els tres casos es normalitzen els valors entre 0 i 1.

Distàncies entre sets, tenim 4 quadrants, enfrontant blogs i científic, com que veure-ho a pel és molt molest, podem fer que calculi la mitjana de cada quadrant i veure.

IMPORTANT: aquesta mètrica s'ha eliminat del conjunt de proves, ja que en fer l'automatització l'algorisme es comportava molt malament en termes de temps de computació. S'afegeixen resultats de proves que es van fer.

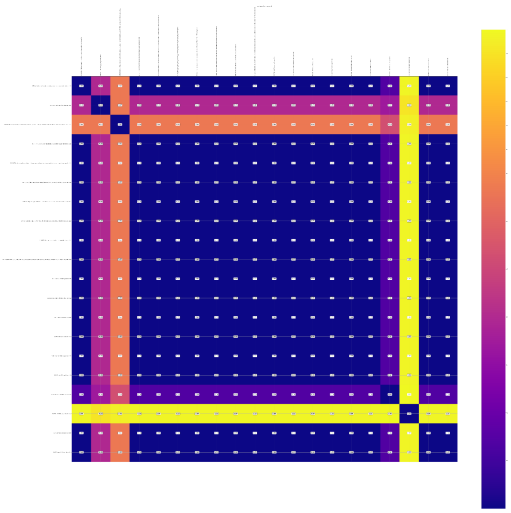

Figura 70: matriu similaritat 20 documents Doc2Vec-Blogs mètrica Word Mover Distance.

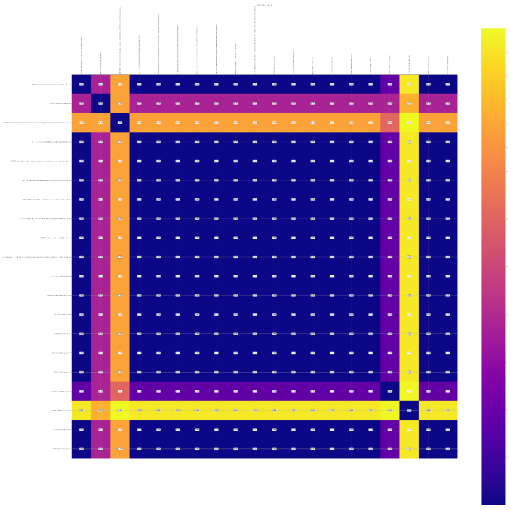

Figura 71: matriu similaritat 20 documents Word2Vec-Blogs mètrica Word Mover Distance.

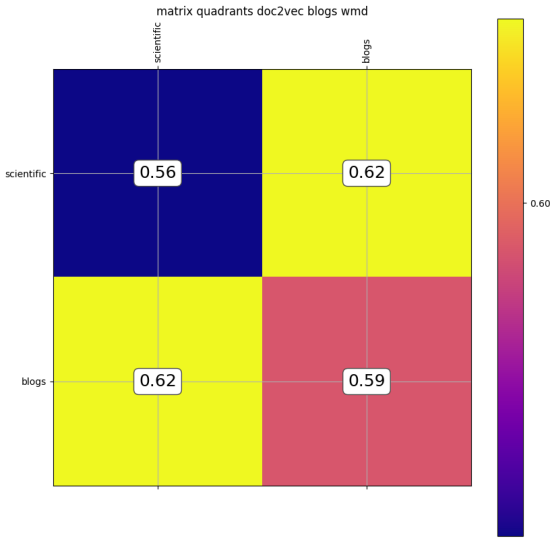

Figura 72: matriu similaritat per classes (300 documents) Doc2Vec-Blogs mètrica Word Mover Distance.

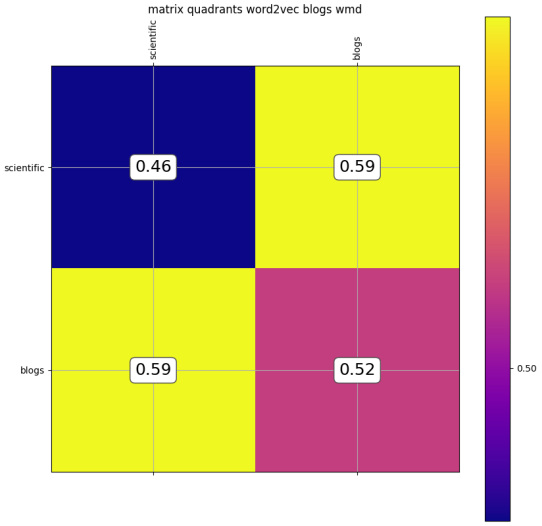

Figura 73: matriu similaritat per classes (300 documents) Word2Vec-Blogs mètrica Word Mover Distance.

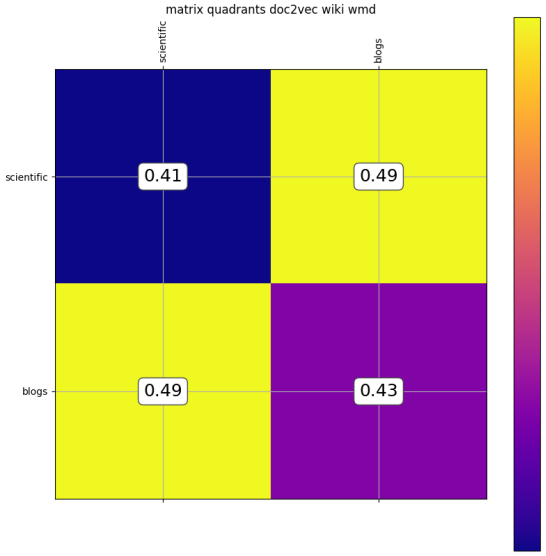

Figura 74: matriu similaritat per classes (300 documents) Doc2Vec-Wikipedia mètrica Word Mover Distance.

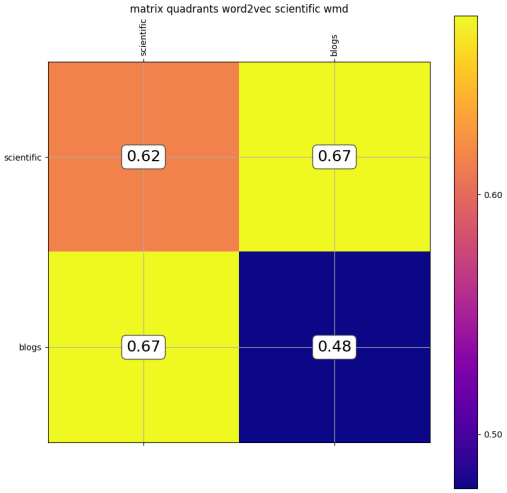

Figura 75: matriu similaritat per classes (300 documents) Word2Vec-Científic mètrica Word Mover Distance.

Podem observar com en cap cas ens ha funcionat bé aquesta mètrica, l'algorisme fa el que s'ha descrit a l'inici de l'apartat, per calcular aquesta mètrica el propi model fa servir directament llistats de *tokens* que representen els documents. S'hauria de mirar amb detall perquè no ha funcionat bé en el nostre cas. En tot cas, per les proves automàtiques és inviable executar un *tfidf* de molts documents i després fer el *WMD*.

Es va parlar de fer el *Relaxed Word Mover Distance*, però per temps no es va arribar a fer, aquest és molt més ràpid en temps de computació i ens permetria poder incloure aquesta mètrica a l'automatització de les proves.

12.3.4. Mitjana de distàncies entre documents del mateix context

Comparem n documents dos a dos, tindrem n^2 comparacions, la premisa és que hi hagi un ordre entre les comparacions, el que volem és disseccionar en quatre quadrants obtenint la mitja de totes les comparacions entre els documents del mateix tema (quadrant *top-left* i per separat quadrant *bottom-right*), per altre banda tindrem els altres dos quadrants que mesuraran la mitja de les distàncies entre documents de diferents temes, aquests dos quadrants donaran el mateix.

La idea d'aquest experiment és que podem veure un nombre elevat de comparacions entre documents sense tindre que revisar un a un.

Precisament aquesta idea és la que hem fet servir pel concepte de quadrants, lo que al ja estar estandarditzada a tot el projecte s'ha fet servir abans de comentar-la.

Comentem una mica com està implementat:

A l'hora d'obtenir els documents el que fem és obtindre els documents però amb les classes, tenim un tipus *List<(String, List<String>, String)>*, on la tercera posició de la tupla és la classe del document.

A l'hora de calcular la matriu de distàncies, hem d'incloure també la classe, com que fem servir com a tipus un *Any* a una de les tuples a l'hora de fer tots els càlculs dels *embeddings* no es suposa cap problema (és com un *payload* que li afegim a tot càlcul).

Quan tenim tots aquests càlculs ens queda una estructura de tipus *List<((String, String), (String, String), float)>* que ens simbolitza nombre del document i classe, el *float* ens indica la distància entre aquests dos. Només ens queda agrupar per classes i fer el càlcul de la mitjana de les distàncies de cada classe. Els noms dels fitxers en aquest cas els ometem.

Fet això, tot lo altre és exactament igual ja que compta com un fitxer de matriu de *csv*.

12.4. Estructura de proves combinatòries

S'ha plantejat una estructura combinatòria on s'executen múltiples testos amb les màximes combinacions possibles. La idea és abastar les màximes combinacions possibles per tal de poder veure una mostra molt gran.

Per poder aconseguir això s'ha fet un projecte que és bastant escalable, fent que afegir una nova combinació és molt senzill.

Tot l'esmentat aquí s'ha explicat en anteriors seccions i l'escalabilitat es comenta a l'apartat d'iteracions.

Tenim 4 categories:

- Context
- Tipus de model
- Mètrica
- Tècnica

Cada prova constarà d'un context escollit, un tipus de model, una mètrica i una tècnica, tot junt formaran un test combinatori.

També, cada test constarà sempre de **x** documents del context científic i **x** documents del context de blogs, en igual proporcions. La **x** només varia a la tècnica escollida. I sempre la primera mitja part de tots el conjunt de documents serà científic i la segona de blogs, així podrem veure millor la similaritat si la mirem en conjunt.

Per exemple, un test combinatori seria context=Scientific, tipus model=word2vec, mètrica=euclidean i tècnica=nxnrows

Cada test combinatori ens donarà una matriu de similiaritat on podrem analitzar com s'ha comportat aquest test.

Més endavant per tal de poder tindre igualtat de condicions en els tests es va implementar un *TSNE* del *embedding* abans d'aplicar la mètrica, això és degut al fet que existeixen mètriques que només contemplen dues dimensions per fer el càlcul. En tot cas, en aquests primers resultats no es contempla *TSNE*, evidentment tampoc s'afegeixen els resultats imatge a imatge.

12.4.1. Resultats

Els resultats els tenim amb *JSON* i hem fet servir l'*script* que vam dissenyar per tal de fer una conversió a imatge amb mapa de calor, ja que avaluar els *JSON* un per un és impossible, s'ha fet visualment en aquesta prova (més endavant veurem que també automatitzem millor el procés).

Com que hi ha tantes proves fetes, el que faré és fixar-me en els quadrants en mitges i classificar en 3 grups:

- (A) Màxima diferència entre grups diferents. El més proper a allò que hauria de ser.
- (B) Diferència no notable.
- (C) Desajustat (grups que no s'haurien d'assemblar, s'assemblen)

En cas d'haver-hi dubtes, ens fixarem també als resultats més específics de ***nxn*** *rows*.

12.4.1.1. Doc2vec context blogs

Doc2Vec Context Blogs				
Mètrica	A	B	C	Comentaris addicionals
Euclidean			■	Essent context blogs igualment dóna una diferència totalment desajustada.
Cosine		■		
Chebyshev			■	Científic amb científic és més llunyà que científic amb blogs.
Manhattan			■	
WMD		■		
Haversine	■			És normal que dins el context de blogs hi hagi més diferència entre blogs que entre científics: millor comportament.
Hamming			■	Potser també es podria encasellar al grup B, en tot cas, és context de blogs i entre documents de blogs funcionarà millor segurament.
Minkowsky p=2			■	
Minkowsky p=3			■	
Minkowsky p=4			■	
Minkowsky p=8			■	
Minkowsky p=16			■	Aquí la diferència s'ajusta més, contra més gran l'hiperparàmetre més es marquen les distàncies.

Taula 7: taula de resultats de l'estructura de proves combinatòries per Doc2Vec-Blogs.

12.4.1.2. Doc2vec context científic

Doc2Vec Context Científic				
Mètrica	A	B	C	Comentaris addicionals
Euclidean				*
Cosine				No ha reaccionat del tot malament tenint en compte el context.
Chebyshev				*
Manhattan				*
WMD				Tot el contrari a allò que hauria de passar, actua malament entre documents científics estant dins del context científic.
Haversine				Molt positiu aquest resultat, molta diferència entre grups de documents diferents i poca entre documents similars: millor comportament.
Hamming				
Minkowsky p=2				*
Minkowsky p=3				*
Minkowsky p=4				*
Minkowsky p=8				*
Minkowsky p=16				Aquí s'accentua una mica la diferència entre documents de tipus blogs, és positiu, però només és una dècima.

Taula 8: taula de resultats de l'estructura de proves combinatòries per Doc2Vec-Científic.

*Massa propera blogs a blogs amb científic

12.4.1.3. Doc2vec context wikipedia

Doc2Vec Context Wikipedia				
Mètrica	A	B	C	Comentaris addicionals
Euclidean				
Cosine				Es comporta bastant bé.
Chebyshev				
Manhattan				*
WMD				*
Haversine				Molt bon comportament: millor comportament.
Hamming				Diferències on haurien d'estar però són mínimes.
Minkowsky p=2				*
Minkowsky p=3				*
Minkowsky p=4				*
Minkowsky p=8				*
Minkowsky p=16				Les diferències s'han anat fent més petites a mesura que anàvem augmentant el paràmetre fent que s'acostin massa entre els quadrants.

Taula 9: taula de resultats de l'estructura de proves combinatòries per Doc2Vec-Wikipedia.

*No tan bé com *Cosine* i *Haversine*, però raonable.

Sembla que cap prova s'ha comportat malament en aquest context i model.

12.4.1.4. Word2vec context blogs

Word2Vec Context Blogs				
Mètrica	A	B	C	Comentaris addicionals
Euclidean				*
Cosine				*
Chebyshev				*
Manhattan				
WMD				
Haversine				
Hamming				No molta distància entre quadrants, encara que les diferències són correctes.
Minkowsky p=2				
Minkowsky p=3				
Minkowsky p=4				
Minkowsky p=8				
Minkowsky p=16				No canvia molt els resultats a mesura que anem augmentant el paràmetre.

Taula 10: taula de resultats de l'estructura de proves combinatòries per Word2Vec-Blogs.

* Molt bon comportament: millor comportament

12.4.1.5. Word2vec context scientific

Word2Vec Context Científic				
Mètrica	A	B	C	Comentaris addicionals
Euclidean				Comportament normal, diferència entre blogs molt petita.
Cosine				A blogs la diferència és molt petita.
Chebyshev				Molt bon comportament.
Manhattan				Raonable.
WMD				No tanta diferència.
Haversine				Diferències similars entre docs de blogs i entre docs de científics: millor comportament.
Hamming				Poca diferència.
Minkowsky p=2				
Minkowsky p=3				Millor comportament que amb p=2, si augmentem pitjor comportament, sembla que és un mínim local.
Minkowsky p=4				
Minkowsky p=8				
Minkowsky p=16				Contra més augmentem pitjor comportament.

Taula 11: taula de resultats de l'estructura de proves combinatòries per Word2Vec-Científic.

S'ha observat que es comporta millor entre documents de blogs que entre documents científics.

12.4.1.6. Word2vec context wikipedia

Word2Vec Context Wikipedia				
Mètrica	A	B	C	Comentaris addicionals
Euclidean				
Cosine				Entre els documents que toca les diferències són correctes, però totes en general estan a la baixa.
Chebyshev				
Manhattan				
WMD				
Haversine				Comportament ideal, poca diferència entre documents similars i molta diferència entre documents diferents.: millor comportament.
Hamming				
Minkowsky p=2				
Minkowsky p=3				
Minkowsky p=4				
Minkowsky p=8				
Minkowsky p=16				En aquest cas és el que s'ha comportat millor dels minkowski.

Taula 12: taula de resultats de l'estructura de proves combinatòries per Word2Vec-Wikipedia.

Notes:

- No hi consta *hamming*, era inviable computar-lo per aquesta combinació de paràmetres.
- Els documents emprats a tots els altres models i contextos han sigut 600 en aquest cas han sigut 300, ja que el temps de computació era excessiu, això és degut al fet que a l'hora d'entrenar el model (aquest es va extreure d'internet ja entrenat) van fer servir vectors de mida molt gran, llavors a l'hora de fer els embeddings ens donen vectors de dimensions molt altes.

12.4.2. Valoració general per quadrants

Semblà que *Word2Vec* es comporta millor en general, almenys en aquests tres contexts amb els documents que hem fet servir.

La prova amb *WMD* és la que pitjor s'ha comportat, juntament amb *Hamming* que en general també ha sigut deficient.

Com que són tantes proves, anem a fer una petita assignació de punts per mètode emprat, -1 punt si has estat al grup C, 0 punts si has estat al grup B, 1 punt si has estat al grup A i 2 punts si has sigut el millor del grup A:

Valoració general								
Mètrica	1	2	3	4	5	6	Punt.	Comentaris
Euclidean	C	B	B	A	A	A	2	
Cosine	B	A	A	A	A	A	5	
Chebyshev	C	B	B	A	A	A	2	
Manhattan	C	B	A	A	A	A	3	
WMD	B	C	A	B	B	B	0	
Haversine	A+	A+	A+	A	A+	A+	13	Millor comportament
Hamming	C	C	B	B	B	?	-2	Comptem com a 0 el ?
Minkowsky p=2	C	B	A	A	A	A	3	
Minkowsky p=3	C	B	A	A	A	A	3	
Minkowsky p=4	C	B	A	A	A	A	3	
Minkowsky p=8	C	B	A	A	A	A	3	
Minkowsky p=16	C	B	B	A	A	A	2	

Taula 13: taula de resultats de l'estructura de proves combinatòries valoració general.

1 → Doc2Vec Context Blogs
2 → Doc2Vec Context Científic
3 → Doc2Vec Context Wikipedia
4 → Word2Vec Context Blogs
5 → Word2Vec Context Científic
6 → Word2Vec Context Wikipedia

Com veiem *Haversine* s'ha comportat molt bé en general tenint les diferències més coherents i ajustades.

Cosine tampoc s'ha comportat del tot malament, *Manhattan* i *Minkowski* amb un paràmetre d'entre 2 i 8 també s'ha comportat bé.

12.5. Prova de perplexitat

Per tal de poder valorar correctament totes les mètriques amb les mateixes condicions, s'ha implementat una reducció de dimensionalitat a dos components (dues dimensions), per tal de poder avaluar *Haversine* amb tots (ja que en aquesta mètrica és imprescindible fer la mesura en dues dimensions).

Aquesta prova que s'explica en aquest apartat s'ha creat amb el propòsit d'escollir una perplexitat per la prova final (que inclourà múltiples proves i arribarem a una conclusió arran d'aquestes).

Volem ajustar una perplexitat general, s'ha provat amb diverses perplexitats, concretament s'han provat valors de perplexitat: **{1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196}** (de l'1 al 14 amb funció **x*x**)

La prova s'ha fet sobre el producte cartesià de:

- Tipus Model
- Context
- Mètrica

Amb el mètode *NxN Rows* per tema de temps de processament, ja que fer una prova amb el mètode de grups pot consumir massa temps, ja de per si, fer una execució consumeix unes 12 hores (trigaríem diversos dies per fer el càlcul), no té massa sentit, aleshores, ho fem per *NxN Rows* que consumeix un temps molt més petit i avaluem en general, amb quina perplexitat s'ha comportat millor i escollim aquesta.

Per tant, hem d'assumir que la prova és amb una mostra relativament petita, hem agafat 10 documents científics i 10 documents de blogs, al final el que volem saber és amb quin valor de perplexitat es comporta millor, com farem tantes proves agregades la mostra és suficient.

Evidentment, aquesta tasca ha de ser una mica automàtica, farem servir tota l'arquitectura de proves combinatòries que tenim feta, aquestes matrius estan exportades en format *JSON* codificades en matriu de similaritat normalitzada (valors d'entre 0 i 1), amb aquests arxius tenim un script que genera imatges amb mapes de calor.

Resultats a l'annex (secció: Prova Perplexitat, resultats plans).

Amb aquests resultats el que s'ha fet és agregar-los per perplexitat, fent l'agregació per perplexitat, i hem distingit 4 categories dins de cada perplexitat: màxims, mínims, mitjanes i variàncies.

12.5.1. Generació dels primers resultats

Els primers resultats que s'han generat han sigut mitjançant *Python* tradicional, s'ha executat un test exhaustiu amb un bucle a un nivell superior de perplexitat, en el següent *script* es pot veure millor:

```
def execute_perplexity_test():
   folder = "execute_perplexity_test" + get_string_timestamp_now()

   for x in range(1, 15):
       folder_nxnrows = os.path.join(folder, "perplexity_nxnrows_" +
str(x*x))
       for m in ModelType:
           for c in Context:
               for l in Metric:
                   if l != Metric.RWMD and l != Metric.WMD and l !=
Metric.HAMMING:
                       if l == Metric.MINKOWSKI:
                           exec_routine(Method.NXNROWS, m, c, l,
folder_nxnrows, 20, 2, tsne_activated=True, components=2,
perplexity=x*x)
                           exec_routine(Method.NXNROWS, m, c, l,
folder_nxnrows, 20, 3, tsne_activated=True, components=2,
perplexity=x*x)
                           exec_routine(Method.NXNROWS, m, c, l,
folder_nxnrows, 20, 4, tsne_activated=True, components=2,
perplexity=x*x)
                           exec_routine(Method.NXNROWS, m, c, l,
folder_nxnrows, 20, 8, tsne_activated=True, components=2,
perplexity=x*x)
                           exec_routine(Method.NXNROWS, m, c, l,
folder_nxnrows, 20, 16, tsne_activated=True, components=2,
perplexity=x*x)
                       else:
                           exec_routine(Method.NXNROWS, m, c, l,
folder_nxnrows, 20, tsne_activated=True, components=2, perplexity=x*x)
```

Hem omès *RMWD*, *WMD* i *Hamming*, les raons estan explicades a un apartat anterior.

Com veiem, això ens exporta una gran quantitat de resultats, exactament 15 carpetes amb 60 fitxers *JSON* cada carpeta. Analitzar un a un és evidentment una pèrdua de temps. S'ha optat llavors a començar a fer servir fulls de *Jupyter Notebook* per tal d'agilitzar aquest procés.

L'estructura del *JSON* és una matriu de distàncies entre els documents agafats i una ristra de paràmetres.

Parlem de distàncies en tot aquest apartat.

12.5.2. Funcionament de l'script

No s'explicaran absolutament tots els passos que conté l'script ni s'entrarà molt en profunditat en aquest, si es vol consultar aquest script, s'ha d'anar a l'annex.

En termes bàsics, el que es vol arran de totes les carpetes i fitxers generats, és tindre un *dataframe* de la llibreria *Pandas* on podem discriminar per perplexitat.

Cada fitxer *JSON* conté una matriu de similaritat i els paràmetres amb els quals s'ha generat aquest fitxer, el primer pas és llegir aquest fitxer i tindre una estructura de dades a memòria amb aquesta estructura, hem fet servir els diccionaris que són els anàlegs als fitxers *JSON*.

Una vegada tenim això, hem de transformar aquesta matriu a màxims, mínims, mitjanes i variàncies per quadrants. Abans d'això hem de definir els noms, els fitxers de categoria científica els anomenarem A i els fitxers de categoria blogs els anomenarem B, per tant, els quadrants queden de la següent manera:

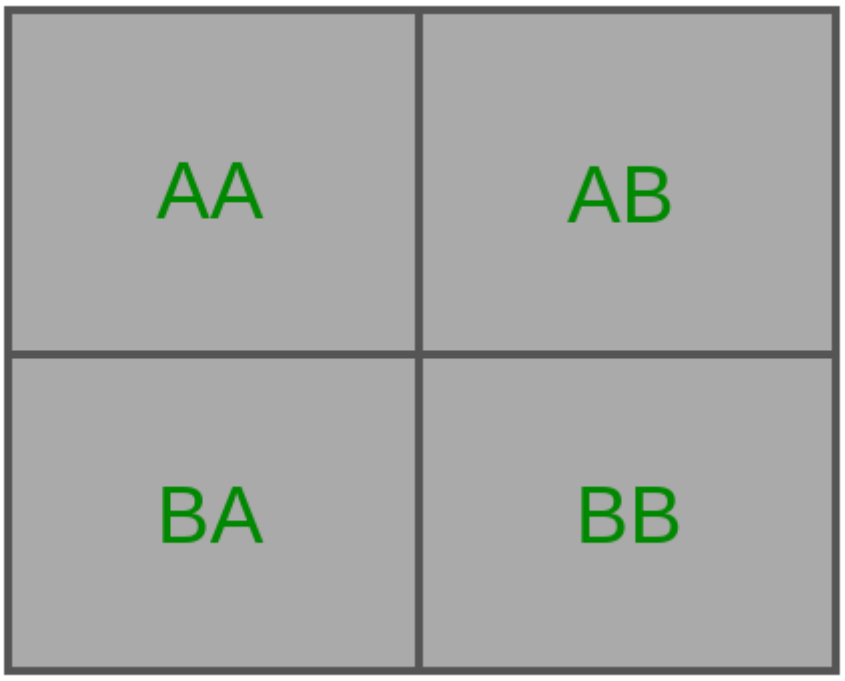

Figura 76: assignació de noms pels quadrants de la matriu de similaritat per classes.

És exactament el mateix a l'assaig de proves amb classes, però calculant les mitjanes nosaltres, amb el mètode per classes només tenim la mitjana, en fer-ho amb una granularitat de document podem fer màxims, mínims, mitjanes, variància o el que se'ns acudeixi.

Per tant, la nostra segmentació suggerida és crear les següents segmentacions per poder avaluar amb més detall, el grup BA no ens interessa, ja que és equivalent al AB.

Grup AA: mínim, màxim, variància i mitjana.
Grup AB: mínim, màxim, variància i mitjana.

Grup BB: mínim, màxim, variància i mitjana.

Tenint això, combinem tot en un diccionari més gran i creem l'estructura de dades de *Pandas dataframe*.

Tenim les següents columnes per cada fitxer:

- minAA
- maxAA
- varianceAA
- meanAA
- minAB
- maxAB
- varianceAB
- meanAB
- minBB
- maxBB
- varianceBB
- meanBB
- method -> sempre serà NXNROWS
- metric
- model
- context
- perplexity
- file

A l'annex es deixa un exemple d'*output* fent un print del *dataframe* (exemple dataframe prova perplexitat).

Per tant, tenint això ja ben estructurat podem començar a fer gràfics més elaborats.

12.5.3. Gràfiques

El plantejament que hem volgut fer és veure els quatre valors principals com s'han comportat de mitjana. Això no vol dir mostrar només la dada mitjana que hem calculat, vol dir que per cada paràmetre (recordem, màxim, mínim, variància i mitjana), cada quadrant i cada perplexitat obtenir una mitjana.

NO s'han omès els 0 de les diagonals, ja que no estem comparant AA contra AB contra BB, sinó que estem comparant perplexitats, per tant, és irrellevant que les mitges d'AA i BB quedin molt per sota que les de AB.

Segons el paràmetre i el quadrant explicarem en cada cas què és el idoni i com s'ha comportat.

12.5.3.1. Paràmetre màximes

En aquest apartat veurem com s'ha comportat el paràmetre de màximes, la premissa que han de seguir és que el quadrant AA i BB sigui baix i el quadrant AB sigui alt.

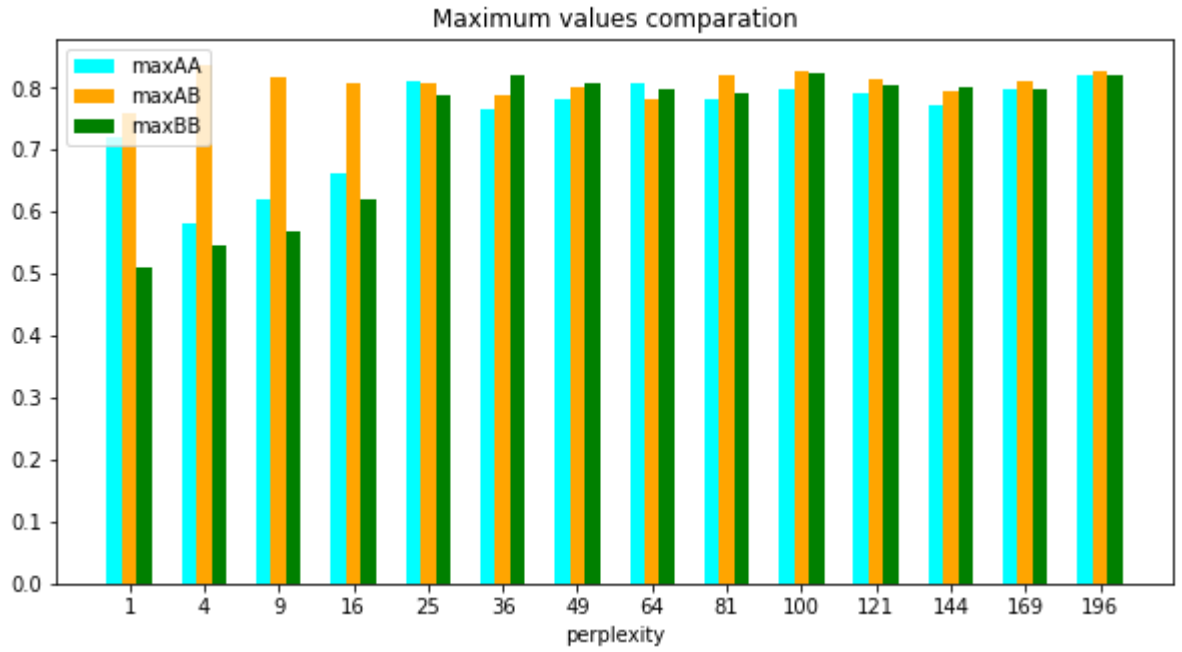

Figura 77: comparació de perplexitats per valors màxims per quadrants.

Evidentment, descartem perplexitats 25, 36, 49, 64, 81, 100, 121, 144, 169, 196 ja que estan molt igualats els quadrants i les altres perplexitats veiem millors diferències.

De les que ens queden, veiem que la perplexitat 1 té una mitjana de màxims del quadrant AA molt similar a la del quadrant AB, i molt baixa respecte al quadrant BB.

La perplexitat 16 la mitjana global és bastant més elevada que les de la 4 i 9, cosa que no és el idoni tampoc, ja que vol dir que s'apropen molt AA, AB i BB.

Les més equilibrades que veiem en aquest gràfic són 4 i 9, són les que respecten millor les premisses comentades a l'inici de l'apartat, si hagués d'escollir, em quedaria amb la 4, pel fet que és la que té els màxims més elevats i els mínims més baixos de mitja.

12.5.3.2. Paràmetre mínimes

La premissa ideal és que el màxim valor sigui entre comparacions del quadrant AB i el mínim entre comparacions del quadrant AA i BB.

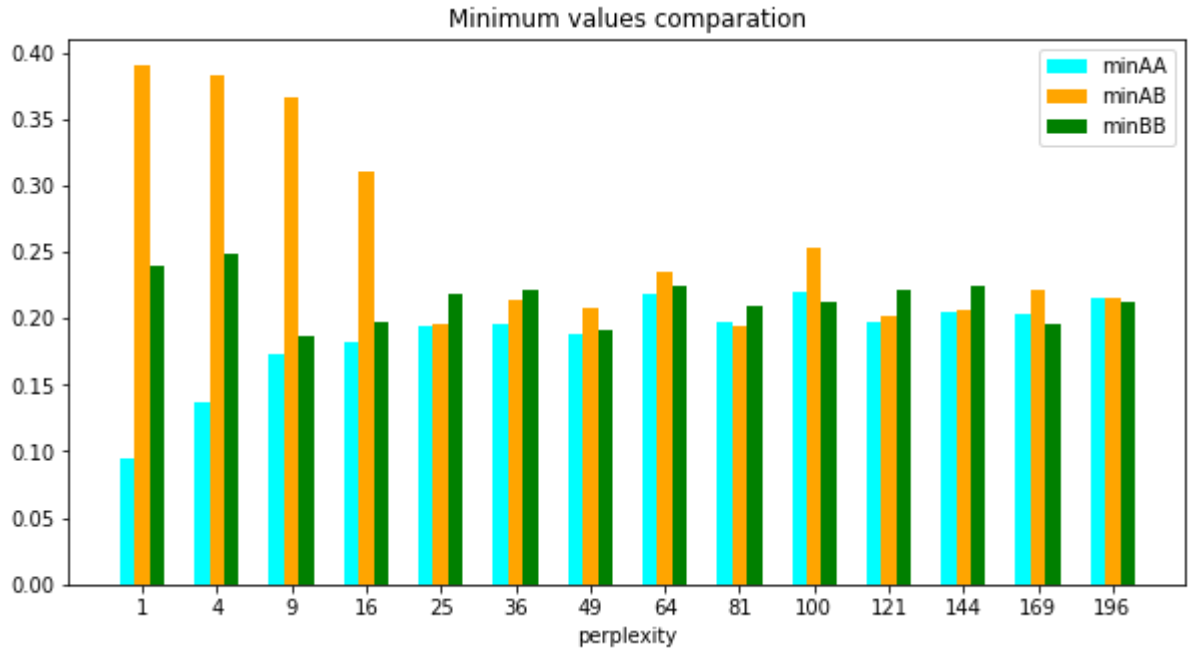

Figura 78: comparació de perplexitats per valors mínims per quadrants.

Descartem perplexitats 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, són el contrari a les premisses especificades.

La perplexitat 1 té un mínim d'AA molt baix, un mínim de BB més alt i un mínim d'AB molt alt, la perplexitat 4 és el mateix, però una mica més anivellat, la 9 es veu que té el millor ràtio AA AB BB, ja que AA i BB són molt similars i cap a la baixa respecte AB i AB és molt alt; i la 16 és equivalent a la 9 però en proporcions més petites.

Si hagués que fer un *ranking* escolliria per ordre: 9, 4, 16, 1.

12.5.3.3. Paràmetre variància

La premissa ideal és que la variància sigui el més petit possible a cada quadrant, ja que això fa que les distàncies siguin més estables i fiables.

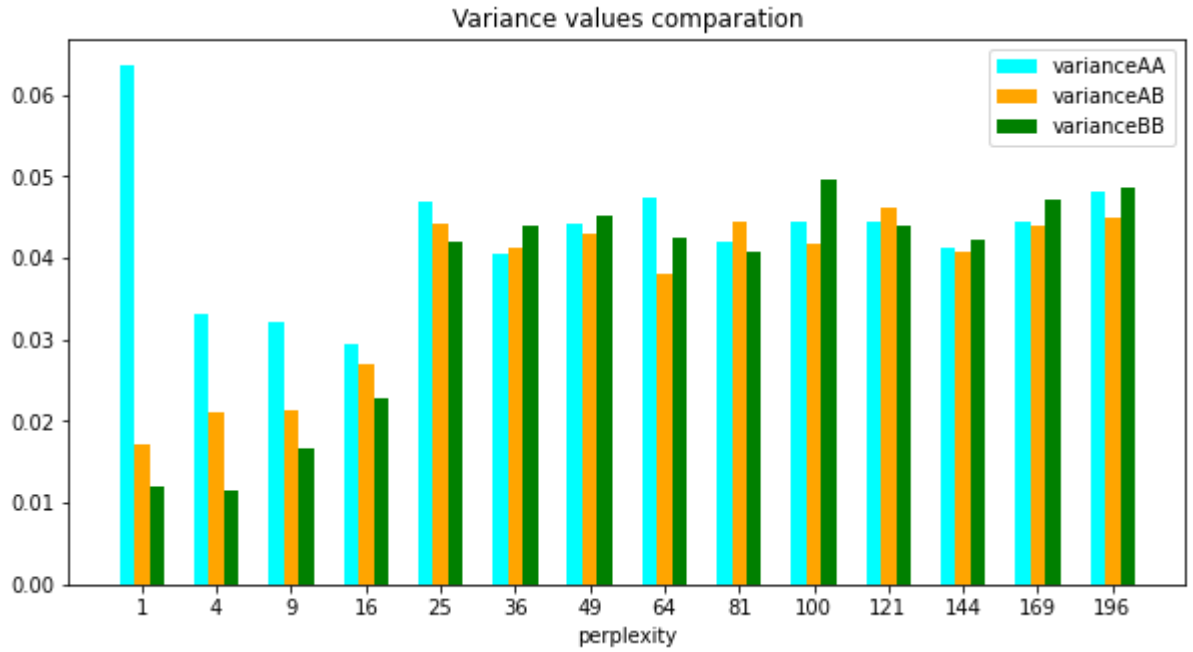

Figura 79: comparació de perplexitats per variància per quadrants.

Descartem perplexitats 25, 26, 49, 64, 81, 100, 121, 144, 169 i 196, en mitja són bastant més elevades a la resta.

Descartem també la perplexitat 1, ja que pel quadrant AA ha crescut molt.

Observem que pel quadrant AA la variància és quasi sempre més elevada que la resta, això és degut al fet que els punts dels documents científics hem vist que quasi sempre són més dispersos que els de blogs.

Ens quedem doncs amb la 4, 9 i 16.

12.5.3.4. Paràmetre mitjana

La premissa ideal és que la mitjana d'AA i BB sigui tan petita com sigui possible i la mitjana AB molt diferenciada cap a un nivell superior respecte AA i BB.

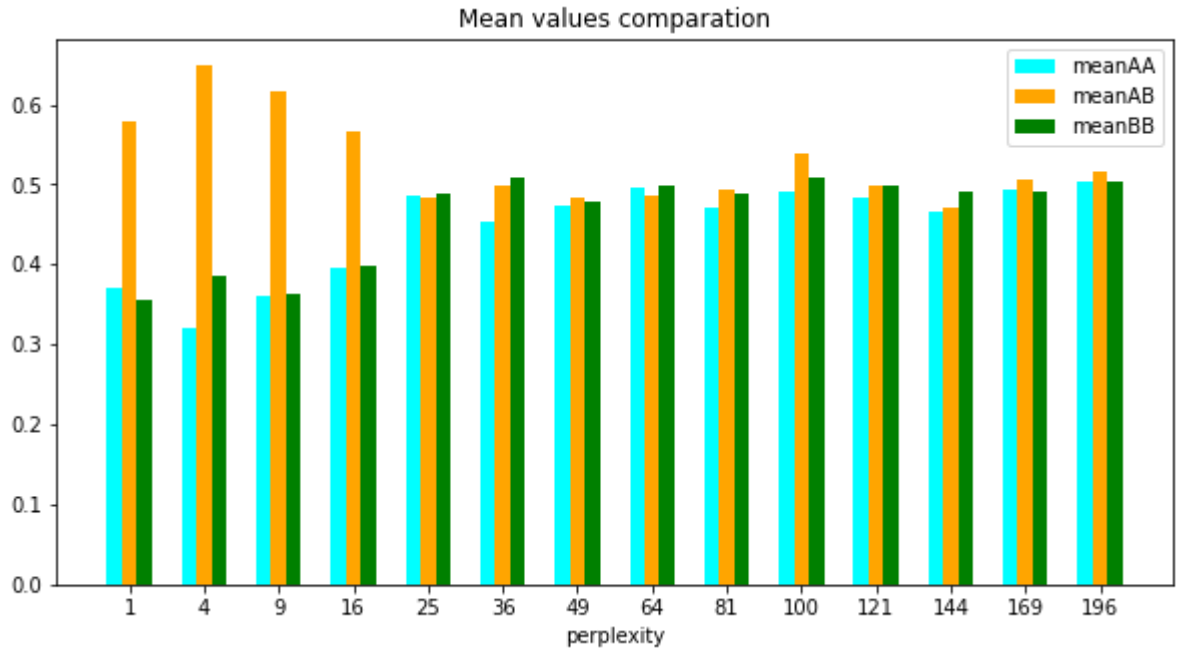

Figura 80: comparació de perplexitats per mitjanes per quadrants.

Descartem perplexitats 25, 26, 49, 64, 81, 100, 121, 144, 169 i 196, ja que no hi ha diferències entre quadrants.

Les quatre perplexitats que ens queden compleixen bastant bé la premissa donada, la que més s'ajusta a aquesta és la perplexitat 4, té el AB més alt i AA BB molt més baixos en proporció, de fet dels més baixos.

Per tant, podem escollir la perplexitat 4 d'aquí. En ordre de millor resultat: 4, 9, 1, 16.

12.5.4. Conclusions

Hem vist que no funcionen gaire bé les perplexitats 25, 36, 49, 64, 81, 100, 121, 144, 169 i 196 en cap de les 4 representacions que hem donat.

Fem recopilació, en ordre de millor comportament descendent:

Màxims: 4, 9
Mínims: 9, 4, 16, 1
Variància: 4, 9, 16
Mitjana: 4, 9, 1, 16

Podem confirmar llavors que la perplexitat 4 és la que s'ha comportat millor en general i és la que escollirem a partir d'ara per fer tota reducció de dimensionalitat.

12.6. Proves finals amb Perplexitat 4

Els assaigs que anem a fer ara ja tots es donen amb perplexitat 4. El primer que hem de fer és generar una ristra de resultats per poder analitzar-los. Per fer això repetirem l'execució, però escollint aquesta perplexitat, es detalla l'execució de l'script:

```
def execute_test_by_groups_perplexity_4():
   folder = "execute_test_by_groups_perplexity_4_" + get_string_timestamp_now()
   for m in ModelType:
       for c in Context:
           for l in Metric:
               if l != Metric.RWMD and l != Metric.WMD and l != Metric.HAMMING:
                   if l == Metric.MINKOWSKI:
                       exec_routine(Method.NXNROWS, m, c, l, folder, 150, 2,
tsne_activated=True, components=2,perplexity=4)
                       exec_routine(Method.NXNROWS, m, c, l, folder, 150, 3,
tsne_activated=True, components=2,perplexity=4)
                       exec_routine(Method.NXNROWS, m, c, l, folder, 150, 4,
tsne_activated=True, components=2,perplexity=4)
                       exec_routine(Method.NXNROWS, m, c, l, folder, 150, 8,
tsne_activated=True, components=2,perplexity=4)
                       exec_routine(Method.NXNROWS, m, c, l, folder, 150, 16,
tsne_activated=True, components=2,perplexity=4)
                   else:
                       exec_routine(Method.NXNROWS, m, c, l, folder, 150,
tsne_activated=True, components=2,perplexity=4)
```

Com veiem, aquesta prova l'hem executat amb 150 documents per combinació, això vol dir que per cada fitxer amb una mètrica, context i tipus de model tindrem un *JSON* de 150^2 comparacions i els paràmetres amb què s'ha executat.

La mostra que hem fet servir en aquest test és molt més gran i en totes les execucions hem fet servir el mètode *NXNRows* com a l'apartat anterior per tal de poder fer una anàlisi més exhaustiu.

12.6.1. Funcionament de l'script

El funcionament és similar al de la prova de perplexitat, és exactament igual fins a crear el *dataframe* del *Pandas* que és el punt de partida que ens interessa tenir per, a partir d'aquí poder fer diferents gràfics discriminant entre mètriques, models o contextos.

També podem fer subgrups discriminatoris, per exemple, volem avaluar els grups que siguin *Doc2Vec* i a més s'hagi fet servir la mètrica de distància euclídea.

Per fer-ho hem creat funcions que ens pinten tres tipus de gràfics (tots tres són scripts que et generen aquests gràfics):

12.6.1.1. Per Coeficients

A aquest gràfic volem veure les dades d'un conjunt discriminatiu en concret, ens mostra la mitjana de *max*, *min*, *mean* i *var* (màxim, mínim, mitjana i variància) extret dels quadrants corresponents. La idea és veure la diferència entre aquests, si el valor és molt elevat vol dir que funciona bé la mètrica, ja que els valors d'AB sempre han de ser menors que els d'AA i els d'AB menors que els de BB, ja que AB són mesures entre diferents tipus de documents i AA i BB entre aquests, llavors la distància indica com de diferents o similars són.

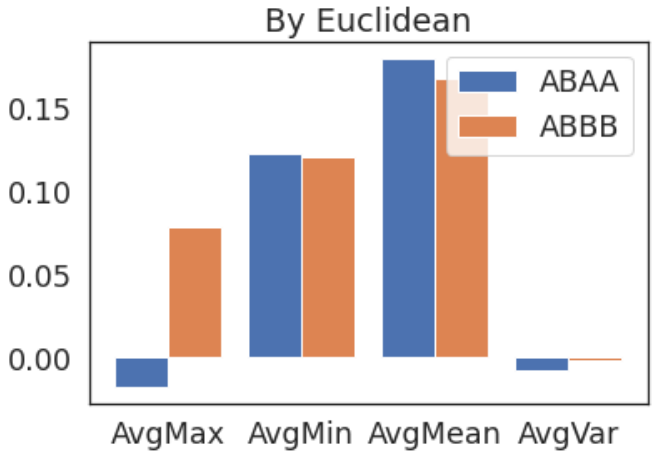

Figura 81: exemple dels diferents tipus de valors per diferències entre els quadrants.

12.6.1.2. Per grups

Aquesta visualització fem servir el mateix criteri a l'hora de mostrar les dades AB - AA i AB - BB, però seleccionant entre *max*, *min*, *mean* i *var*; i escollint una columna discriminant.

Per exemple, en el gràfic que s'ha adjuntat a continuació:

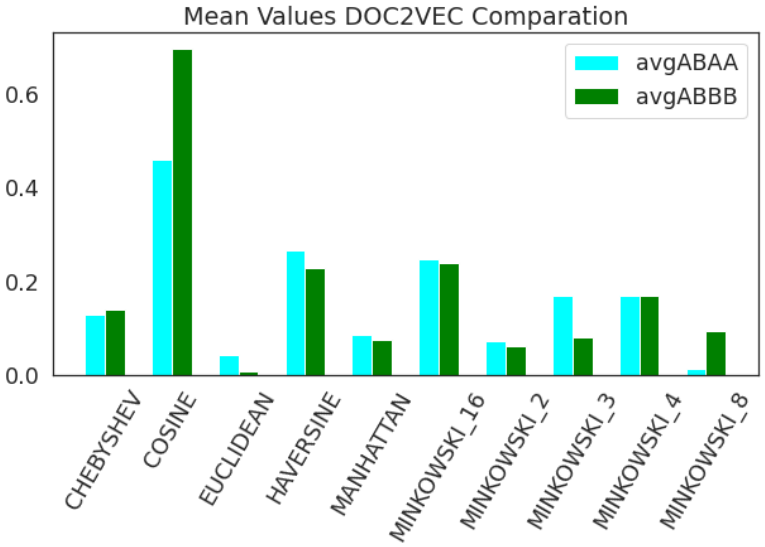

Figura 82: exemple de mitjanes de les diferents mètriques discriminant per Doc2Vec per diferències entre els quadrants.

El que veiem és una discriminació per la columna mètrica, amb la dada de *mean* i mirant per cada columna el diferencial mitjà entre AB i AA; i AB i BB, en aquest cas hem agafat un grup pare que és *Doc2Vec*.

També es podria no agafar un pare i discriminar per exemple per model:

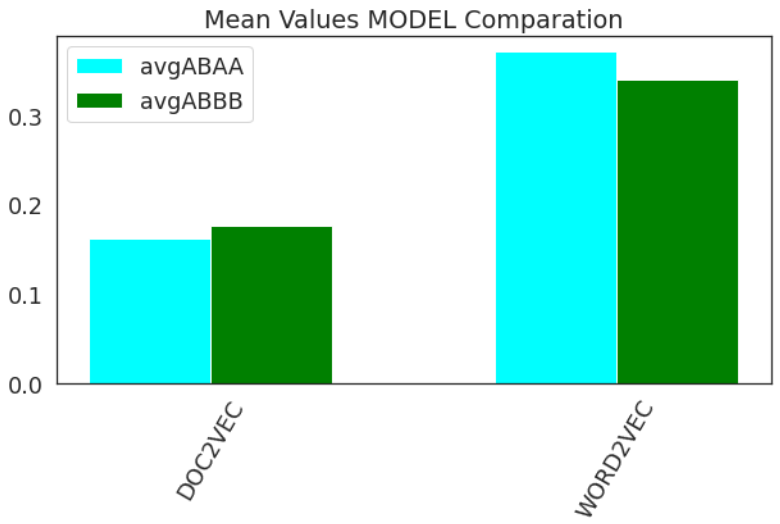

Figura 83: exemple de mitjanes dels dos models sense discriminar per diferències entre els quadrants.

O bé podem agafar un pare que només tingui *Doc2Vec* i la mètrica euclidiana i discriminar per context per màxims:

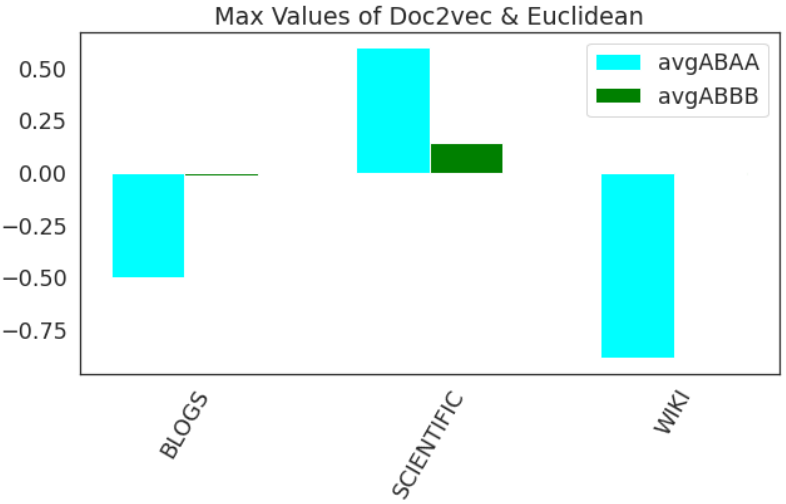

Figura 84: exemple dels màxims valors dels tres context discriminat pel model Doc2Vec i la mètrica Euclidiana per diferències entre els quadrants.

Aquest script ens dóna molta flexibilitat a l'hora de poder agafar dades agregades o més específiques.

12.6.1.3. Per grups isolats

És similar a l'anterior, però no es fa diferència entre quadrants sinó els quadrants directament, la relació entre les etiquetes és:

- AA -> SC
- AB -> Mixed
- BB -> Blogs

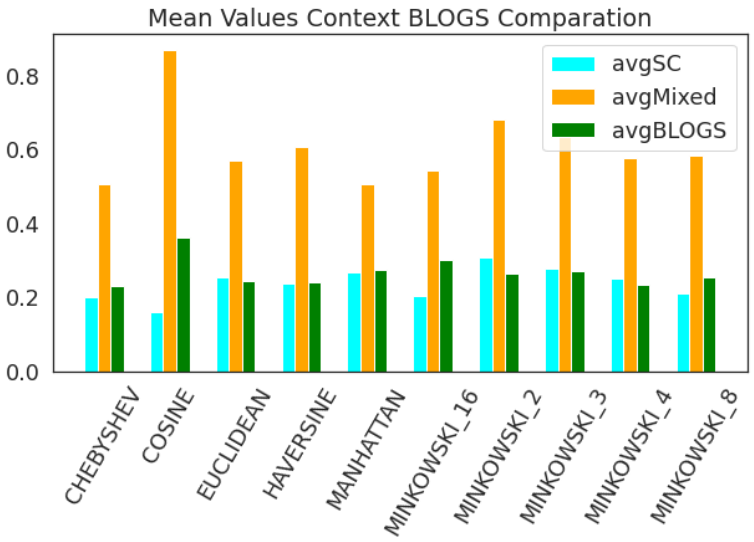

Figura 85: exemple dels valors mitjans de totes les mètriques discriminant pel context Blogs per quadrants.

Tots els scripts que generen aquests gràfics són a l'annex.

12.6.1.4. Conceptes bàsics sobre la parametrització

Definirem uns conceptes bàsics abans de procedir. Ens referim a parametrització estadística als mínims, màxims, mitjanes i variàncies. Quan fem un gràfic de l'estil (adjuntem un que ja hem fet servir):

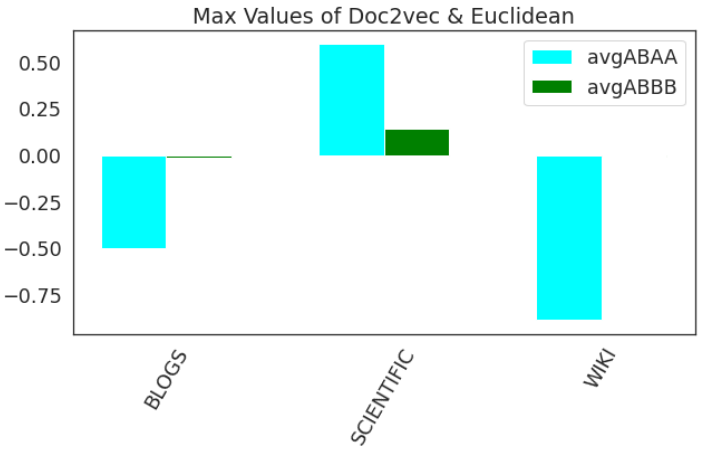

Figura 86: mateixa figura que la figura 84.

A aquest gràfic estem dient que tots els arxius que componen els subgrups escollits, de cada fitxer i cada quadrant s'ha agafat el màxim valor i s'ha fet la mitjana sobre els màxims de tots els valors obtinguts per quadrants.

El mateix amb els mínims agrupats, de tots els mínims que componen els grups pintats es mostren per mitjana. Amb la variància igual, de cada fitxer s'ha agafat la variància per quadrants i s'ha fet la mitjana de totes aquestes dels grups mostrats. I la mitjana que es mostra per paràmetre compon les mitjanes de cada quadrant de cada fitxer.

12.6.2. Gràfiques

A aquesta secció visualitzarem diferents gràfics segmentats per diferents columnes del *dataset*. La idea és intentar agafar alguna noció de per on es comporta millor.

12.6.2.1. Generals per paràmetre

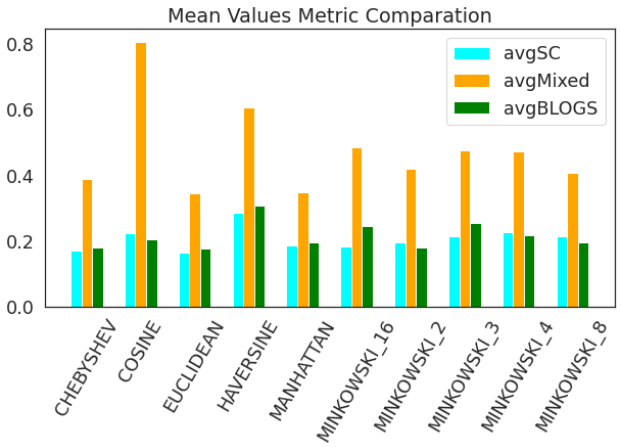

Figura 87: valors mitjans de les mètriques per quadrants.

Veiem que hi ha diverses diferències com per exemple que *cosine similarity* és la que millor es comporta en general amb *Mixed* molt alt i SC i Blogs relativament baixos, sí que és cert que en general totes les mètriques respecten la proporció esmentada.

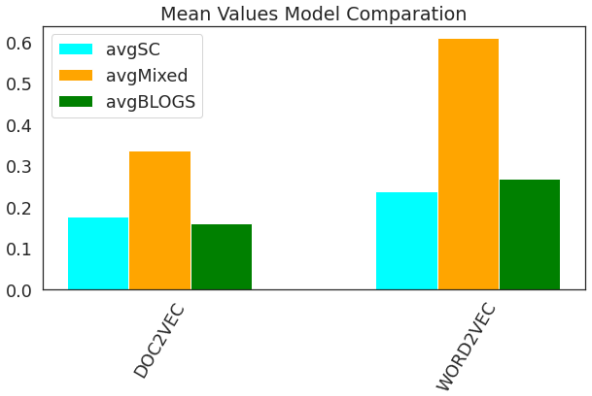

Figura 88: valors mitjans dels dos models per quadrants.

En aquest gràfic veiem com els valors de distàncies de *Doc2Vec* són inferiors als de *Word2Vec*, adjuntem el següent gràfic de distàncies de la mateixa agrupació de dades:

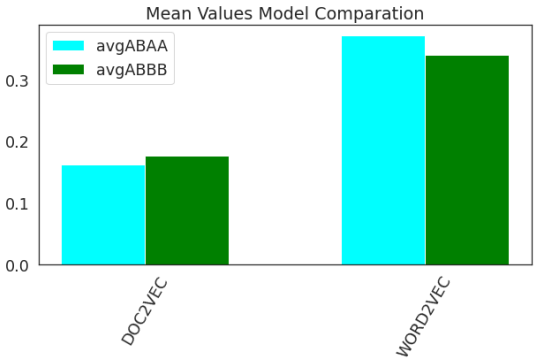

Figura 89: Valors mitjans dels dos models per diferències de quadrants.

Aproximadament els valors són la meitat, això té certes implicacions, però veurem amb més detall els grups més petits dins de *Doc2Vec* per veure si alguna mètrica flaqueja molt.

Adjuntem màxims, mínims i variància:

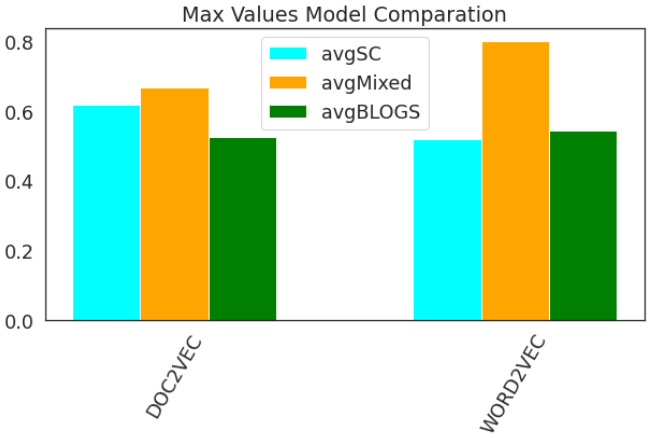

Figura 90: valors mitjans dels dos models per quadrants.

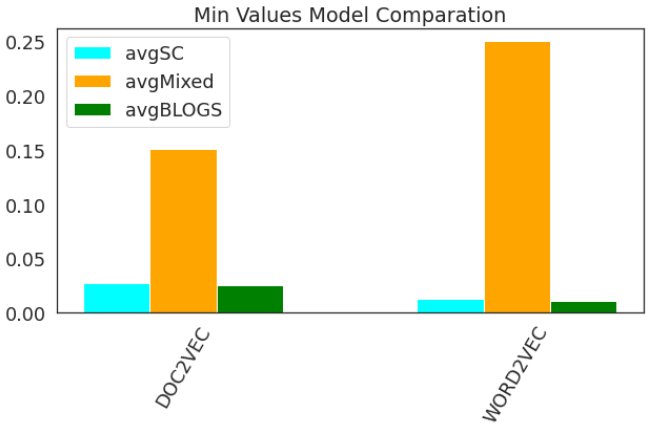

Figura 91: valors mínims dels dos models per quadrants.

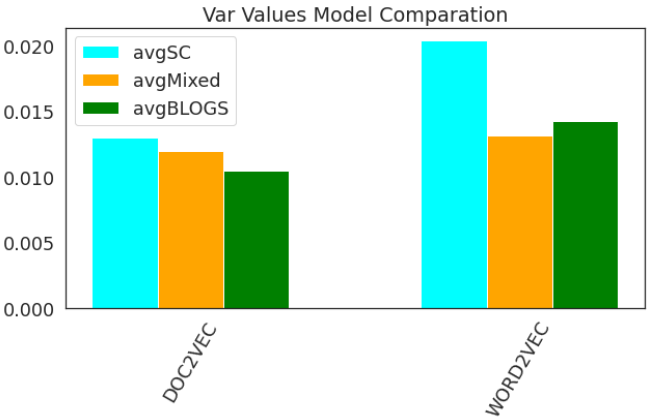

Figura 92: variàncies dels dos models per quadrants.

Veiem com els màxims s'ajusten millor a *Word2Vec* encara que la diferència entre ells no és massa, però respecta bé la premissa Mixed > SC, Blogs.

Als mínims també, en els dos gràfics aquestes dues mètriques es comporten relativament bé, més *Word2Vec* que *Doc2Vec* també. La variància és més petita a *Doc2Vec* pel fet que al gràfic de mitjanes hem vist valors més grans per W*ord2Vec*, podem veure també al SC de *Word2Vec* amb molta variància, això és degut al fet que quan vam visualitzar el núvol de punts dels documents vèiem que estaven molt dispersos els punts de SC.

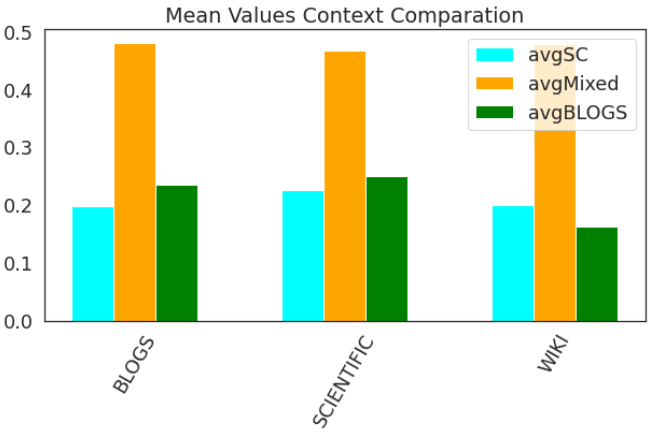

Figura 93: valors mitjans dels tres contextos per quadrants.

Els tres contextos s'han comportat bastant bé, el que millor Wikipedia.

12.6.2.2. Per mètriques

Mètriques per models i mètriques per context (5 gràfics), les generals ja les hem exposat

Primer exposem les mètriques per models (blogs, científic i wikipedia):

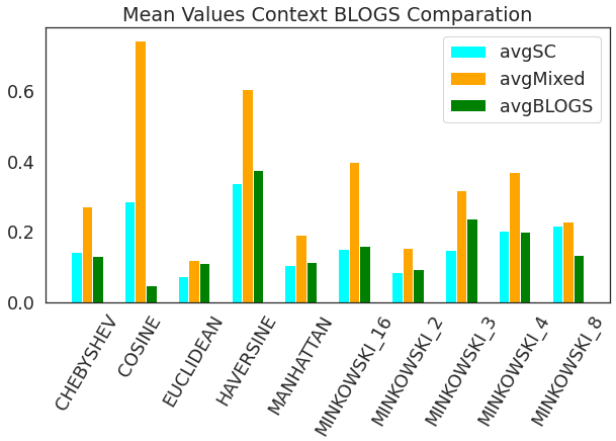

Figura 94: valors mitjans de les mètriques discriminant pel context Blogs per quadrants.

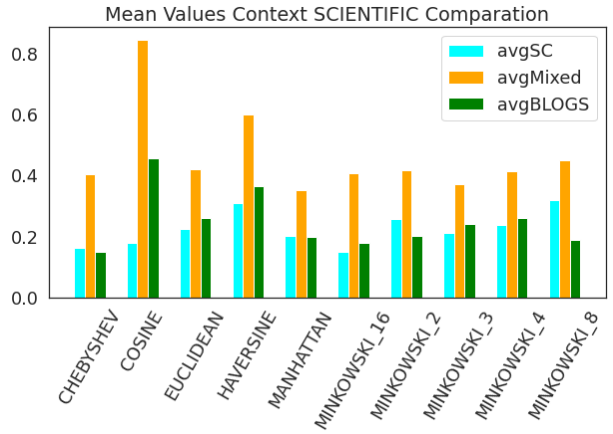

Figura 95: valors mitjans de les mètriques discriminant pel context Científic per quadrants.

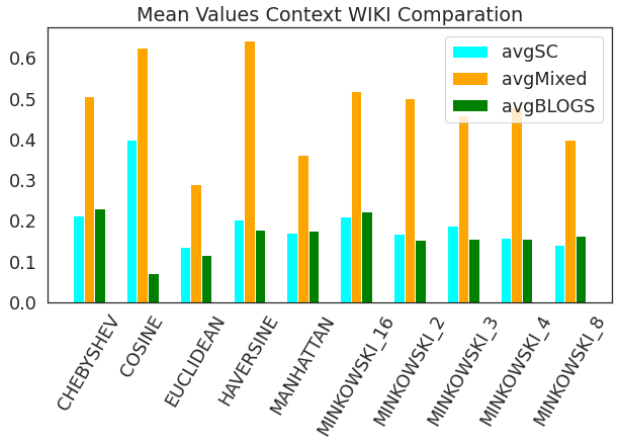

Figura 96: valors mitjans de les mètriques discriminant pel context Wikipedia per quadrants.

Podem veure com la mètrica *cosine similarity* funciona molt bé per documents mixtos, però quan comparem documents científics penalitza molt la distància que s'obté dels *embeddings*.

La mètrica euclidiana és la que pitjor ha funcionat en general.

Haversine té uns valors molt acceptables encara que al context de Blogs no s'ha comportat del tot bé.

També fer l'avaluació al context de blogs ens dona pitjors resultats en general que a qualsevol altre context, encara que amb *cosine similarity* els resultats són magnífics.

En general, observem que al context de Blogs és on tenim pitjors resultats de mitjana, després a Científic una mica millor i a Wikipedia en general bé.

I ara exposarem les mètriques per model:

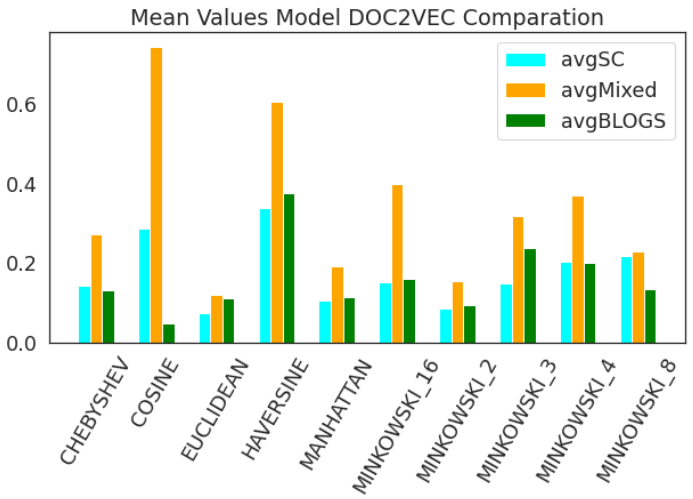

Figura 97: valors mitjans de les mètriques discriminant per Doc2Vec per quadrants.

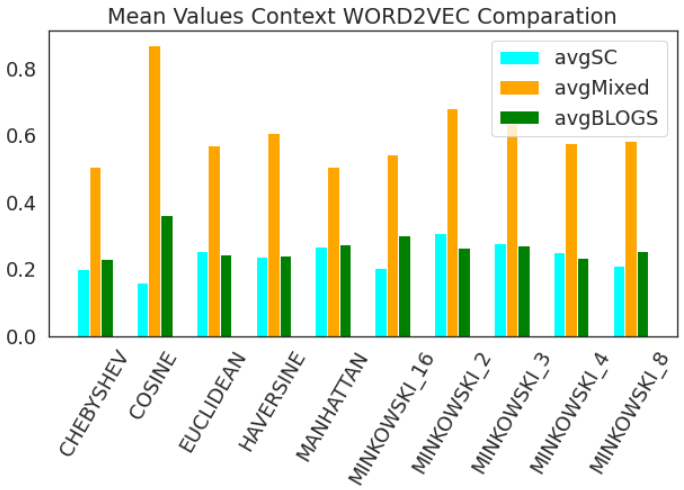

Figura 98: valors mitjans de les mètriques discriminant per Word2Vec per quadrants.

A *Word2Vec* veiem que totes les mètriques són bastant estables i a *Doc2Vec* sí que en veiem bastant diferència de comportament.

A *Doc2Vec* la mètrica euclidiana es comporta molt malament juntament amb la de *Minkowski* amb p = 2, la que millor es comporta i amb bastant diferència és la de *cosine similarity*, *haversine* no té del tot bon comportament, ja que veiem que documents similars (mateix context) dóna una diferència una mica gran i si ho comparem amb el mixed no és molt el diferencial.

Anem a veure la millor i pitjor mètrica fins ara, *cosine* i *euclidean* per màxims, mínims, mitjana i variància:

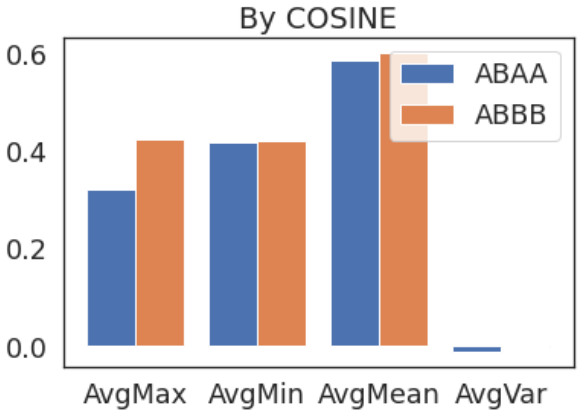

Figura 99: diferents tipus de valors discriminant per Cosine Similarity per diferències entre els quadrants.

Quan el valor és negatiu vol dir que el *Mixed* és més gran que el tipus concret, en aquest cas a *Euclidean* els màxims de SC de mitjana són més grans que els de *Mixed*, a anteriors núvols de punts vam veure que els punts de SC eren bastant esparsos en comparació del de Blogs, en fer comparacions és raonable que ens surti a *Euclidean* que els màxims de SC són majors que els de Mixed, a *Cosine* no ens surt aquest resultat perquè, la manera que tenim de representar la distància amb *Euclidean* és traçar una línia recta entre els dos punts una cosa que fem a simple vista, a *Cosine* és el que es fa és fer-ho mitjançant un angle i veure com de tancat o obert és aquest i amb les coordenades es fa el càlcul, no és tan intuïtiu a simple vista i per això obtenim resultats tan dispars.

En general veiem que les diferències són bastant petites a *Euclidean* i això es tradueix en el fet que *Mixed* més alt però no tant en comparació de SC i Blogs.

12.6.2.3. Per contextos

Exposarem els resultats per contextos disgregats per més endavant poder treure una conclusió respecte als contextos (encara que és impossible determinar si un és millor o pitjor, ja que tenen usos diferents).

Mostrem els resultats sobre els dos models:

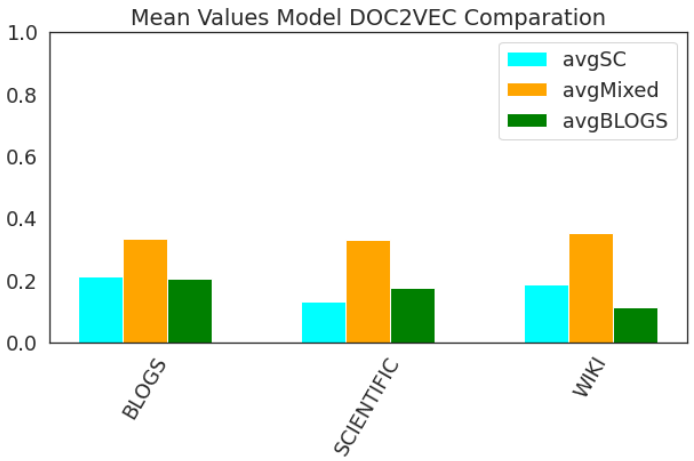

Figura 100: valors mitjans dels diferents contexts discriminant per Doc2Vec per quadrants.

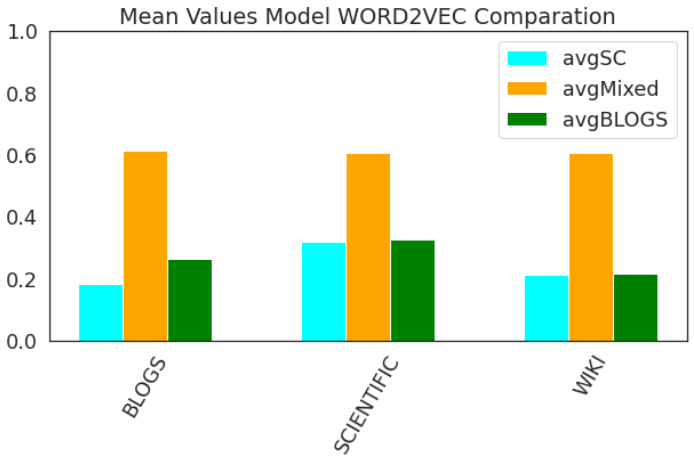

Figura 101: valors mitjans dels diferents contexts discriminant per Word2Vec per quadrants.

Veiem com proporcionalment *Word2Vec* té unes distàncies superiors, és normal trobar aquestes diferències, perquè internament tenen algunes diferències.

El context que millor sembla funcionar en general és el de *Wikipedia*, ens dona uns valors molt raonables dins dels dos models. El context de blogs a *Doc2Vec* no es comporta del tot bé tenint diferències petites entre tipus de documents i el científic a *Word2Vec* és el que pitjor comportament té.

Si només ens fixem en diferencial:

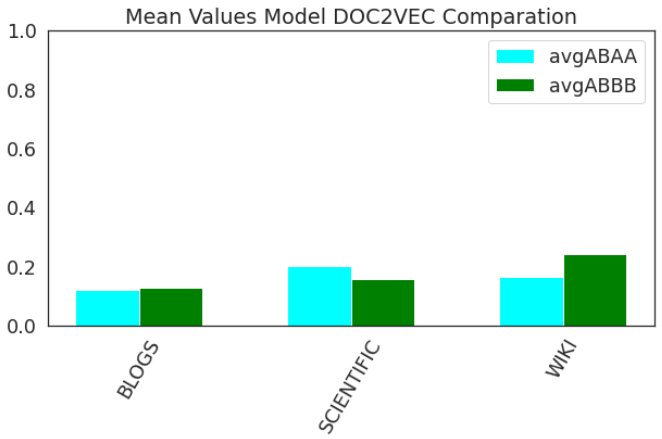

Figura 102: valors mitjans dels diferents contexts discriminant per Doc2Vec per diferències de quadrants.

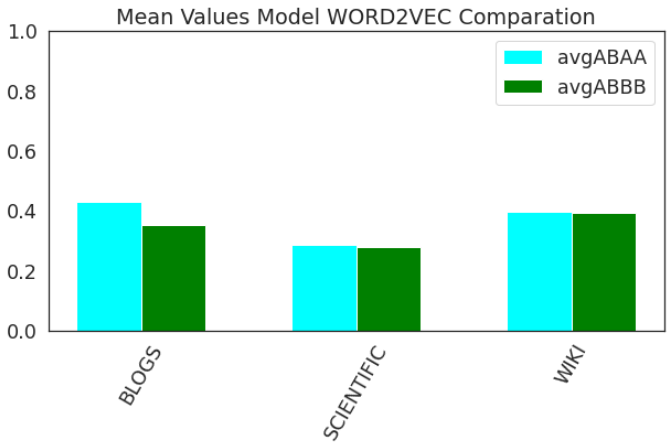

Figura 103: valors mitjans dels diferents contexts discriminant per Word2Vec per diferències de quadrants.

Podem observar el que hem dit millor, Blogs a *Doc2Vec* amb comportament una mica dolent i veiem com a *Word2Vec* wikipedia i blogs tenen un comportament similar.

Ara mostrarem el comportament dels contextos de sis mètriques (s'ometen tots els paràmetres de *Minkowski* menys el p=3 pel fet que es comporten similar):

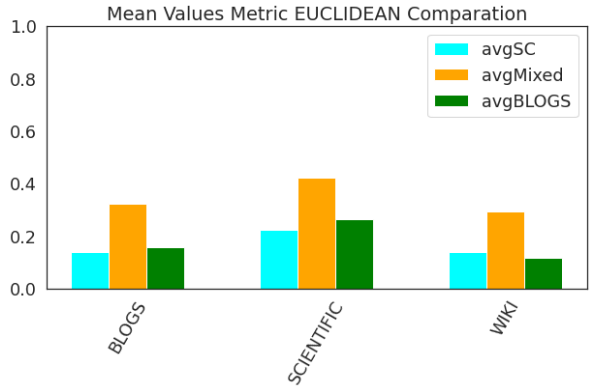

Figura 104: valors mitjans dels diferents contexts discriminant per la mètrica Euclidiana per quadrants.

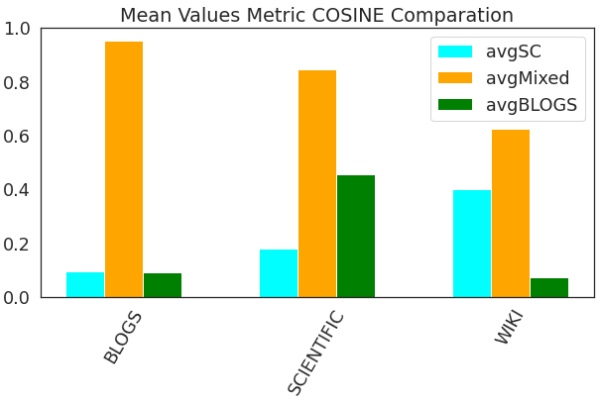

Figura 105: valors mitjans dels diferents contexts discriminant per Cosine Similarity per quadrants.

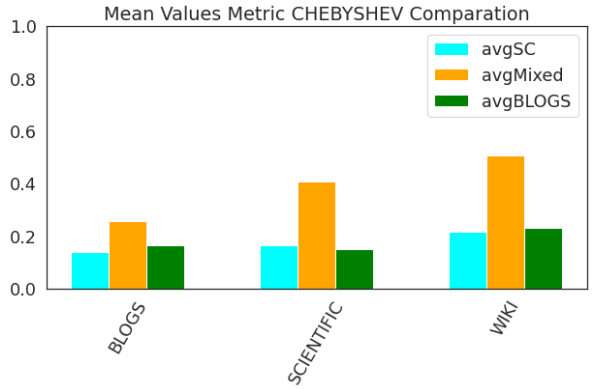

Figura 106: valors mitjans pels diferents contexts discriminant per Chebyshev per quadrants.

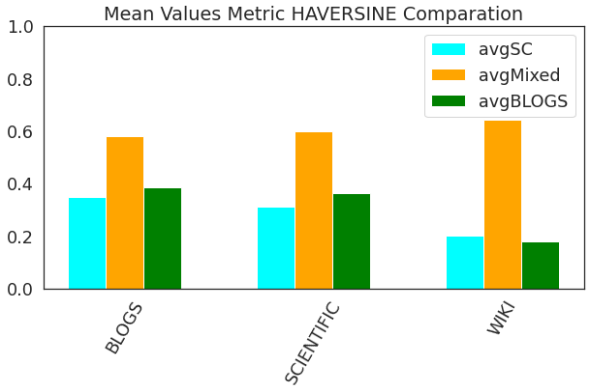

Figura 107: valors mitjans pels diferents contexts discriminant per Haversine per quadrants.

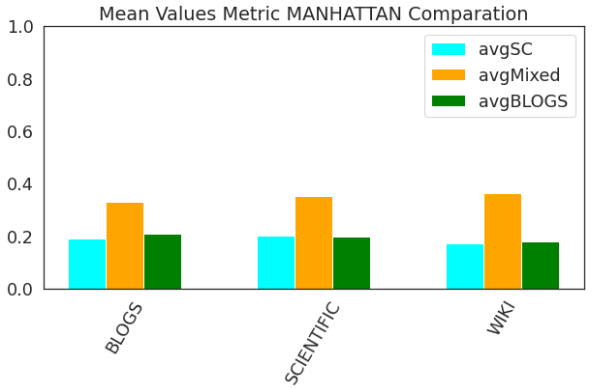

Figura 108: valors mitjans pels diferents contexts discriminant per Manhattan per quadrants.

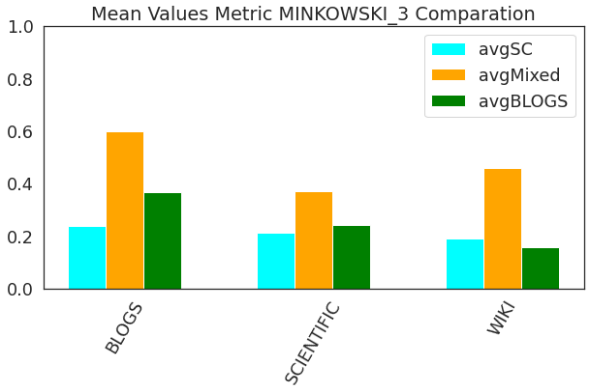

Figura 109: valors mitjans pels diferents contexts discriminant per Minkowski p=3 per quadrants.

Degut com es distribueixen els punts al context de Blogs la mètrica *Cosine* dóna millors resultats en aquest.

Manhattan no té molt bons resultats i no destaca en cap, de fet, té uns resultats quasi idèntics als tres contextos.

Haversine té uns resultats acceptables, encara que la proporcionalitat és superior.

Chebyshev no es comporta gaire bé a Blogs, però sí a cientific i wikipedia.

Haversine no té un diferencial molt alt a blogs i científic, però en wikipedia es comporta bastant bé.

Minkowski amb p=3 té un comportament raonable al context de wikipedia, al context de blogs amb documents de blogs sembla que no es comporta gaire bé, al context científic no té un comportament notable.

13. Conclusions

En aquest projecte hem fet un repàs de tots els conceptes associats en aquest, després hem dedicat una secció completa (procediments) a analitzar i explicar com s'ha dut a terme el projecte explicant també alguns conceptes importants per tal de poder entendre millor el projecte, hem dedicat una secció a com escalar el projecte i finalment hem fet la secció d'assaigs que és on hem analitzat com es comporta cada element del projecte.

Dedicarem un apartat a cada element del projecte mesurable i al final escollirem de cada secció l'element que millor s'ha comportat en general.

13.1. Resum

Hem trobat que de totes les mètriques analitzades, *cosine similarity* és la mètrica que millor es comporta en general, en quasi totes les execucions cosine ha tingut el millor comportament excepte el cas dels models amb el context Wikipedia, que aquí *Haversine* sí que ha tingut un millor comportament, la segona mètrica amb millor comportament ha sigut *Minkowski* amb el paràmetre p=16.

Respecte als models no es pot dir quin model és millor, ja que cadascun té particularitats i es poden fer servir en diferents circumstàncies.

Finalment, els contextos són bastant diferents entre si, el que podem veure és que al context de Wikipedia és on els resultats han sigut més coherents, ja que és el context més global.

13.2. Mètriques

Hem inclòs totes les mètriques que hem proposat al principi encara que no hàgim analitzat totes, en aquesta secció volem fer una valoració per cada mètrica, veure punts forts i punts febles i més tard arribar a una conclusió de quina mètrica és la que millor es comporta.

13.2.1. Minkowski

Aquesta mètrica l'hem analitzat amb diferents valors d'hiperparàmetre i l'hem exposat com mètriques diferents amb **p={2, 3, 4, 8, 16}**, hem agafat aquesta mostra per diversificar una mica els possibles resultats. Hem de tindre en compte que per **p = 0** és equivalent a la mètrica euclidiana i per **p = inf** o **-inf** és la mètrica *Manhattan*.

A *Doc2Vec* no s'ha comportat massa bé, de fet, quan hem incrementat la p és quan millor s'ha comportat en aquest model, per aquest motiu la mètrica Euclidiana ha funcionat tan malament a aquest model.

A *Word2Vec* ha tingut un comportament no massa dispar en variacions del hiperparàmetre, valors diferents han tingut resultats similars.

Quan hem fet la comparativa per contextos, a blogs és on hem vist un pitjor comportament a mesura que el paràmetre era més baix, a p=0 els resultats han sigut catastròfics. Al context de científic ha tingut un millor comportament, però també segueix una mica la tònica de decrementant p disminueix el rendiment. Al context de científic podem veure un comportament similar sigui quin sigui l'hiperparàmetre.

Ens podria resultar que incrementant p cap a l'infinit es millora el rendiment general, però tenim l'exemple de Manhattan que no s'ha comportat massa bé amb un rendiment similar al de *Euclidean*.

Per tant ens quedem amb p=16 si volem el millor cas de la mostra que hem agafat, faltaria veure si altres valors superiors funcionen millor i trobar un màxim local de rendiment.

13.2.2. Euclidean

L'hem comentat una mica a l'anterior secció, però el desgranarem una mica millor.

En general, juntament amb *Manhattan*, ha sigut la mètrica que pitjor s'ha comportat, encara que hi han resultats on *Manhattan* dóna un millor rendiment.

On podem observar el pitjor comportament és al context de Blogs, on els quadrants SC, Blogs i Mixed tenen una distància mitjana molt similar i baixa el que indica que si volem fer-la servir a la pràctica, no tindrem molt bons resultats. Segurament aquest mal comportament a Blogs es degui a com estan distribuïts els punts en aquest context, tal com hem pogut previsualitzar a anteriors seccions.

Si discriminem per model podem veure que a *Doc2Vec* en general es comporta la pitjor amb un comportament similar al context de blogs, si mirem amb *Word2Vec* sí ha tingut un comportament que està dins de la mitja.

Així, doncs, no el faria servir sobretot a *Doc2Vec*, i a *Word2Vec* si no tingués més opció sí que podria treure-li profit.

13.2.3. Cosine similarity

Sense dubte és la que millor s'ha comportat en quasi tots els tests que hem fet.

Si ho mirem per models, a *Doc2Vec* ha tingut el millor comportament amb diferència, el valor mitjà del quadrant mixed molt alt, el valor Blogs molt baix i el de científic relativament baix. A *Word2Vec* no és tan exagerat, però en comparació a les altres mètriques també és el que ha tingut el millor comportament.

Ara, mirant per contextos, resulta que ha respectat bastant bé els tipus de documents comparats segons el context, al context Blogs amb documents científics ha tingut bastant pitjor comportament que al

context Blogs amb documents de blogs, que seria lògic (pitjor comportament comparant documents que no són del mateix context). Al context científic ha passat exactament el mateix, documents científics amb molt bon comportament i documents de blogs amb comportament bastant dolent. Ara, al context de *wikipedia* sí que no ha sigut la mètrica que millor s'ha comportat, les comparacions de documents científics no han sigut gaire bones, les de blogs bastant bones.

En tot cas, podem veure que és la mètrica més fiable respectant les premisses i fins i tot actuant com hauria d'actuar dins dels contextos.

13.2.4. Chebyshev

Aquesta mètrica es pot aplicar multidimensionalment, per tant la reducció de dimensionalitat hagi afectat una mica als resultats.

El comportament d'aquesta mètrica està dins de la mitjana, té un comportament acceptable en general. Si mirem la mètrica per contextos podem veure que amb Blogs és on s'ha comportat pitjor, encara que podríem dir que proporcionalment el comportament ha sigut similar en els 3, una mica millor al context científic.

El mateix cas si ho mirem per models, valors relativament baixos a *Doc2Vec* en comparació de les mètriques que millor s'han comportat, i a *Word2Vec* sembla que està dins de la mitjana el comportament.

No veig un comportament ni especialment bo ni especialment dolent, diferències entre documents Científics i documents de Blogs són molt similars, no és una mala mètrica, però hi ha d'altres amb millor comportament.

13.2.5. Haversine

Aquesta mètrica és només per dues dimensions, ja que mesura distàncies respecte a la superfície d'una esfera, evidentment per calcular aquesta mètrica també s'ha fet una reducció de la dimensionalitat.

El comportament és magnífic al comportament de wikipedia, al context científic no tan bo, pel fet que amb documents similars dóna diferències altes. El mateix al context de Blogs, documents similars amb diferències altes, encara que els valors generals siguin més alts no és un bon comportament.

Per models podem observar que a *Doc2Vec* tenim un cas similar, valors alts de diferències en els tres quadrants, el qual no ens indica en absolut un bon comportament. Respecte *Word2Vec* està dins de la mitja.

Si hagués de triar una mètrica, no faria servir aquesta, ja que em proporciona dades no molt bones, només me les ha proporcionat bones al context de Wikipedia.

13.2.6. Manhattan

Aquesta mètrica l'hem comentada una mica a *Minkwoski*, ja que és un cas particular d'aquesta. És de les que pitjor s'ha comportat en general, sigui per contextos o sigui per models.

No la faria servir quasi en cap cas, s'hauria de veure si en algun cas molt concret dóna uns bons resultats, suposo que en una distribució de punts molt particular podria, per exemple punts ubicats com si fos una quadrícula.

13.2.7. Word Mover Distance

Aquesta mètrica no l'hem fet servir pels motius esmentats abans, el càlcul era molt costós computacionalment i la variant més lleugera no es va poder incloure a la planificació per temps.

13.2.8. Hamming

No s'ha inclòs dins dels testos, ja que primer, no donava en cap cas bons resultats, les diferències entre quadrants eren mínimes i això no és un indicatiu de bon comportament, i segon, computacionalment era molt costós de calcular amb la implementació que vàrem fer, si es volgués incloure en aquests testos, s'hauria de trobar una implementació amb un cost computacional més petit.

13.2.9. Valoració general

Farem un *ranking* a ull sobre les gràfiques generades i veurem objectivament qui s'ha comportat millor.

Per ordre descendent de bon comportament:

Blogs: Cosine, Minkowski 16, Minkowski 4, Haversine, Chebyshev, Manhattan, Minkowski 3, Minkowski 2, Minkowski 8, Euclidean.

Científic: Cosine, Chebyshev, Minkowski 16, Haversine, Minkowski 8, Minkowski 4, Manhattan, Euclidean, Minkowski 3, Minkowski 2.

Wikipedia: Haversine, Cosine, Minkowski 16, Chebyshev, Minkowski 2, Minkowski 8, Minkowski 4, Minkowski 3, Manhattan, Euclidean

Doc2Vec: Cosine, Minkowski 16, Haversine, Chebyshev, Minkowski 4, Manhattan, Minkowski 3, Minkowski 2, Minkowski 8, Euclidean.

Word2Vec: Cosine, Minkowski 2, Minkowski 3, Chebyshev, Haversine, Minkowski 4, Minkowski 8, Euclidean, Minkowski 16, Manhattan

General: Cosine, Minkowski 16, Haversine, Chebyshev, Minkowski 3, Minkowski 4, Minkowski 2, Minkowski 8, Manhattan, Euclidean

Recordem que el *ranking* simplement és una manera objectiva de valorar els testos que s'han fet, ja hem esmentat múltiples vegades que cada mètrica té la seva manera de funcionar i dependrà de diferents factors a l'hora d'aplicar-la.

13.3. Models

En aquesta secció avaluarem els resultats obtinguts per cada model que s'ha fet servir al projecte. No hem fet un desenvolupament específic de les gràfiques per models perquè realment les podem veure a través de les altres gràfiques.

Respecte a l'apartat de mètriques a *Doc2Vec* hem vist que és on varien més els resultats, això pot ser degut a com distribueix en l'espai multidimensional els punts, diferents mètriques tindran un comportament diferent, cosa que a *Word2Vec* no passa, a *Word2Vec* tenim uns valors molt semblants entre si excepte *cosine similarity* que té els valors més dispars.

Si ho mirem per contextos podem veure com els valors són bastant més petits en general en comparació a *Word2Vec*, i molt semblants entre si, a científic i wikipedia es poden observar resultats més dispars entre quadrants. A *word2vec*, per altra banda, sí que veiem resultats bastant més elevats, on els quadrants mixed tenen una mitjana quasi idèntica i el context científic és on pitjor s'ha comportat, el que millor al context de wikipedia.

En aquesta secció no es poden decidir guanyadors, ja que són dos models que es basen en un concepte molt semblant però amb diferències internes (explicat a l'apartat corresponent). El que sí que es podria deduir és que ens funciona millor a *Doc2Vec* amb documents *tagged* que sense, veient les diferències que ens han sortit per mètriques.

13.4. Contextos

Finalment, en aquesta secció comentarem el comportament dels diferents contextos que hem fet servir a tot el projecte.

13.4.1. Blogs

Hem d'assumir que aquest context no és molt general igual que el científic, per tant, és normal que puguem veure millors comportaments a wikipedia.

Si ho mirem per models, a *Doc2Vec* no ha tingut molt bon comportament, diferències entre quadrants molt petites, símptoma de mal comportament. A *Word2Vec* sí que veiem un bon comportament, millor que el context científic, encara que les diferències dins del quadrant de blogs són superiors a les de científic.

Si ho mirem per mètriques, a *cosine similarity* ha tingut un rendiment espectacular, amb diferències entre quadrants del mateix tipus molt baixes i mixed molt altes, a totes les altres mètriques el comportament no ha sigut molt bo si ho comparem amb cosine.

Podem concloure doncs que si fem servir context de blogs, els millors resultats els hem obtés amb la mètrica de *cosine* i al context de *Word2Vec*.

13.4.2. Científic

Aquest context recordem que és molt específic, més inclús que el de blogs, ja que agrupa termes molt concrets, ja vam veure al núvol de punts com els grups eren més sòlids i poc esparsos.

A *Doc2Vec* ha tingut un bon comportament i l'esperat, el quadrant de blogs té una mitjana superior al de científic que és l'esperable. En canvi, a *Word2Vec* deixa molt a desitjar, ja que té unes diferències entre quadrants altes, possiblement el fet que a *Doc2Vec* estiguin *taggejats* els documents influeix molt a l'hora de distribuir els punts a l'espai multidimensional.

Si en fixem en la mètrica *cosine similarity* a científic, té el quadrant de blogs amb un valor molt alt i el quadrant de científic molt més baix, és un comportament esperable dins d'un context d'aquest tipus. A les altres mètriques no hi ha molt a destacar, en general els valors són molt estables on els quadrants de blogs i científic tenen uns valors molt similars, si ens fixem, sempre el valor de blogs supera el científic, encara que sigui per poc excepte al *Chebyshev*.

D'aquest context podem concloure que funciona molt bé amb documents del mateix tipus però amb documents d'altres tipus bastant pitjors.

13.4.3. Wikipedia

Aquest context és el més general que tenim, i no comparem documents d'aquest context, sinó que tot document que farem servir no s'ha fet servir per entrenar aquest model, per tant, podem tindre una visió una mica més global de com es comporten els diferents models i mètriques.

A *Doc2Vec* tenim un comportament semblant als altres contextos, però amb uns valors particulars més bons, al context de blogs els valors són els més baixos dels tres contextos, i els de científics sí són una mica més elevats, a més conservant la proporció mixed > blogs, sc. A *Word2Vec* sí que veiem el millor comportament amb diferència, on sc i blogs són els valors més baixos dels tres contextos i un mixed semblant als altres.

En aquest context *cosine* no s'ha comportat gens bé amb els documents científics, però té un comportament molt bo amb els de blogs, una altra mètrica que ha anat molt bé amb aquest context ha sigut *Haversine* que ha resultat amb les diferències entre quadrants més elevades, a *Minkowski* també bastant bé, podem observar com en tots ha tingut un rendiment superior als altres a tota mètrica excepte podríem dir euclidiana.

En conclusió, és notable que és un context molt més global, ja que el rendiment general ha sigut molt millor que als altres contextos, als núvols de punts segurament aïllen bastant bé cada context formant grups diferenciats.

14. Bibliografia

2011. Blog Authorship Corpus - Cohen Courses. http://curtis.ml.cmu.edu/w/courses/index.php/Blog_Authorship_Corpus.

n.d. Understanding UMAP. Accessed April 14, 2022. https://pair-code.github.io/understanding-umap/.

n.d. Understanding UMAP. Accessed April 14, 2022. https://pair-code.github.io/understanding-umap/.

“[1301.3781] Efficient Estimation of Word Representations in Vector Space.” 2013. arXiv. https://arxiv.org/abs/1301.3781.

“The 15 Best Car Subwoofers Under 300$ in 2022: Expert's Top Choices.” n.d. Electronics Market. Accessed April 14, 2022. https://mlexplained.com/2018/09/14/paper-dissected-visualizing-data-using-t-sne-explained/.

Alford, Anthony. 2020. “Google Open-Sources Reformer Efficient Deep-Learning Model.” InfoQ. https://www.infoq.com/news/2020/02/google-reformer-deep-learning/.

“Bag-of-words model.” n.d. Wikipedia. Accessed April 14, 2022. https://en.wikipedia.org/wiki/Bag-of-words_model.

“Beautiful Soup Documentation — Beautiful Soup 4.9.0 documentation.” n.d. Crummy. Accessed April 14, 2022. https://www.crummy.com/software/BeautifulSoup/bs4/doc/.

Beri, Aditya. 2020. “Stemming vs Lemmatization. Truncate a word to its root or base… | by Aditya Beri.” Towards Data Science. https://towardsdatascience.com/stemming-vs-lemmatization-2daddabcb221.

Bohra, Ankur. 2019. “XLNet — SOTA pre-training method that outperforms BERT.” Medium. https://medium.com/logits/xlnet-sota-pre-training-method-that-outperforms-bert-26d4e9978983.

Briggs, James. 2021. “How to Fine-Tune BERT Transformer Python.” Towards Data Science. https://towardsdatascience.com/how-to-train-bert-aaad00533168.

Brownley, Clinton. 2016. “Parsing PDFs in Python with Tika | Clinton Brownley's Decision Analytics.” Clinton Brownley's Decision Analytics. https://cbrownley.wordpress.com/2016/06/26/parsing-pdfs-in-python-with-tika/.

“Chebyshev distance.” n.d. Wikipedia. Accessed April 14, 2022. https://en.wikipedia.org/wiki/Chebyshev_distance.

“Cosine similarity.” n.d. Wikipedia. Accessed April 14, 2022. https://en.wikipedia.org/wiki/Cosine_similarity.

“Distributed Representations of Sentences and Documents.” n.d. Stanford Computer Science. Accessed April 14, 2022. https://cs.stanford.edu/~quocle/paragraph_vector.pdf.

“Distributed Representations of Sentences and Documents.” n.d. Stanford Computer Science. Accessed April 14, 2022. https://cs.stanford.edu/~quocle/paragraph_vector.pdf.

“Distributed Representations of Sentences and Documents.” n.d. Stanford Computer Science. Accessed April 14, 2022. https://cs.stanford.edu/~quocle/paragraph_vector.pdf.

Duque, Tiago. n.d. “How to build a Lemmatizer. And why | by Tiago Duque | Analytics Vidhya.” Medium. Accessed April 14, 2022. https://medium.com/analytics-vidhya/how-to-build-a-lemmatizer-7aeff7a1208c.

Dyson, Paul. n.d. “inflect · PyPI.” PyPI. Accessed April 14, 2022. https://pypi.org/project/inflect/.

“Euclidean distance.” n.d. Wikipedia. Accessed April 14, 2022. https://en.wikipedia.org/wiki/Euclidean_distance.

“From Word Embeddings To Document Distances.” n.d. Proceedings of Machine Learning Research. Accessed April 14, 2022. http://proceedings.mlr.press/v37/kusnerb15.pdf.

“From Word Embeddings To Document Distances.” n.d. Proceedings of Machine Learning Research. Accessed April 14, 2022. http://proceedings.mlr.press/v37/kusnerb15.pdf.

“Gensim: Topic modelling for humans.” 2021. Radim Řehůřek. https://radimrehurek.com/gensim/.

“GloVe: Global Vectors for Word Representation.” n.d. Stanford NLP Group. Accessed April 14, 2022. https://nlp.stanford.edu/pubs/glove.pdf.

“Hamming distance.” n.d. Wikipedia. Accessed April 14, 2022. https://en.wikipedia.org/wiki/Hamming_distance.

"Haversine formula." n.d. Wikipedia. Accessed April 14, 2022. https://en.wikipedia.org/wiki/Haversine_formula.

Horev, Rani. 2018. "BERT Explained: State of the art language model for NLP." Towards Data Science. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

"How can I quickly estimate the distance between two (latitude, longitude) points?" 2013. Stack Overflow. https://stackoverflow.com/questions/15736995/how-can-i-quickly-estimate-the-distance-between-two-latitude-longitude-points.

"How to Use t-SNE Effectively." 2016. Distill.pub. https://distill.pub/2016/misread-tsne/.

"How to Use t-SNE Effectively." 2016. Distill.pub. https://distill.pub/2016/misread-tsne/.

"The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)." 2018. Jay Alammar. http://jalammar.github.io/illustrated-bert/.

"The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)." 2018. Jay Alammar. http://jalammar.github.io/illustrated-bert/.

"Inner product space." n.d. Wikipedia. Accessed April 14, 2022. https://en.wikipedia.org/wiki/Inner_product_space.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://arxiv.org/pdf/1810.04805.pdf.

Jagtap, Rohan. 2020. "XLNet: Autoregressive Pre-Training for Language Understanding | by Rohan Jagtap." Towards Data Science. https://towardsdatascience.com/xlnet-autoregressive-pre-training-for-language-understanding-7ea4e0649710.

"jhlau/doc2vec: Python scripts for training/testing paragraph vectors." n.d. GitHub. Accessed April 14, 2022. https://github.com/jhlau/doc2vec.

Ma, Edward. 2018. “Word Distance between Word Embeddings | by Edward Ma.” Towards Data Science. https://towardsdatascience.com/word-distance-between-word-embeddings-cc3e9cf1d632.

Ma, Edward. 2018. “Word Distance between Word Embeddings | by Edward Ma.” Towards Data Science. https://towardsdatascience.com/word-distance-between-word-embeddings-cc3e9cf1d632.

Ma, Edward. 2018. “Word Distance between Word Embeddings | by Edward Ma.” Towards Data Science. https://towardsdatascience.com/word-distance-between-word-embeddings-cc3e9cf1d632.

“machine-learning-articles/albert-explained-a-lite-bert.md at main · christianversloot/machine-learning-articles.” 2021. GitHub. https://github.com/christianversloot/machine-learning-articles/blob/main/albert-explained-a-lite-bert.md.

“Minkowski distance.” n.d. Wikipedia. Accessed April 14, 2022. https://en.wikipedia.org/wiki/Minkowski_distance.

Mosic, Ranko. n.d. “Reformer: The Efficient Transformer | by Ranko Mosic | Medium.” Ranko Mosic. Accessed April 14, 2022. https://medium.com/@ranko.mosic/reformer-the-efficient-transformer-a2329c8410d1.

“MUSE: Multilingual Unsupervised and Supervised Embeddings - Meta Research.” n.d. Meta Research. Accessed April 14, 2022. https://research.fb.com/downloads/muse-multilingual-unsupervised-and-supervised-embeddings/.

“nlp - Does PorterStemmer supports languages other than english?” 2019. Stack Overflow. https://stackoverflow.com/questions/57026011/does-porterstemmer-supports-languages-other-than-english.

“nltk.corpus package.” n.d. NLTK. Accessed April 14, 2022. https://www.nltk.org/api/nltk.corpus.html.

"nltk.tokenize package." n.d. NLTK. Accessed April 14, 2022. https://www.nltk.org/api/nltk.tokenize.html.

OpenAI. 2020. "Language Models are Few-Shot Learners." Language Models are Few-Shot Learners. https://arxiv.org/pdf/2005.14165.pdf.

"OpenAI GPT-3: Everything You Need to Know." 2021. Springboard. https://www.springboard.com/blog/data-science/machine-learning-gpt-3-open-ai/.

"The Porter stemming algorithm." n.d. Snowball. Accessed April 14, 2022. http://snowball.tartarus.org/algorithms/porter/stemmer.html.

"Preprocess Text in Python --- A Cleaner and Faster Approach." 2019. Think.Data.Science. https://www.thinkdatascience.com/post/preprocess-your-text-for-nlp-models-cleaner/.

"Pretrained Embeddings." n.d. Wikipedia2Vec. Accessed April 14, 2022. https://wikipedia2vec.github.io/wikipedia2vec/pretrained/.

"python - How does spacy lemmatizer works?" 2017. Stack Overflow. https://stackoverflow.com/questions/43795249/how-does-spacy-lemmatizer-works.

"RoBERTa." n.d. GitHub. Accessed April 14, 2022. https://github.com/pytorch/fairseq/tree/main/examples/roberta.

"RoBERTa: An optimized method for pretraining self-supervised NLP systems." n.d. Facebook AI Research. Accessed April 14, 2022. https://ai.facebook.com/blog/roberta-an-optimized-method-for-pretraining-self-supervised-nlp-systems/.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. n.d. "Re-evaluating Word Mover's Distance." Re-evaluating Word Mover's Distance. https://arxiv.org/pdf/2105.14403.pdf.

Shperber, Gidi. 2017. "A gentle introduction to Doc2Vec. TL;DR | by Gidi Shperber | Wisio." Medium. https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e.

"Stemming and lemmatization." n.d. Stanford NLP Group. Accessed April 14, 2022. https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html.

"Taxicab geometry." n.d. Wikipedia. Accessed April 14, 2022. https://en.wikipedia.org/wiki/Taxicab_geometry.

"t-distributed stochastic neighbor embedding." n.d. Wikipedia. Accessed April 14, 2022. https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding.
"t-distributed stochastic neighbor embedding." n.d. Wikipedia. Accessed April 14, 2022. https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding.
Teja, Sai. n.d. "What are Stop Words.How to remove stop words." Medium. Accessed April 14, 2022. https://medium.com/@saitejaponugoti/stop-words-in-nlp-5b248dadad47.

"1301.3781v3 [cs.CL] 7 Sep 2013." 2013. arXiv. https://arxiv.org/pdf/1301.3781.pdf.

"t-SNE." n.d. Interactive Chaos. Accessed April 14, 2022. https://interactivechaos.com/es/manual/tutorial-de-machine-learning/t-sne.

van Kooten, Pascal. n.d. "contractions · PyPI." PyPI. Accessed April 14, 2022. https://pypi.org/project/contractions/.

"Visual Analysis of Research Paper Collections Using Normalized Relative Compression." 2019. Semantic Scholar. https://www.mdpi.com/1099-4300/21/6/612/pdf.

"What is the advantage of choosing ASCII encoding over UTF-8?" 2011. Software Engineering Stack Exchange. https://softwareengineering.stackexchange.com/questions/97247/what-is-the-advantage-of-choosing-ascii-encoding-over-utf-8.

"What is the difference between lemmatization vs stemming?" n.d. Stack Overflow. Accessed April 14, 2022. https://stackoverflow.com/questions/1787110/what-is-the-difference-between-lemmatization-vs-stemming.

"Word2Vec Tutorial - The Skip-Gram Model · Chris McCormick." 2016. Chris McCormick. http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/.

Xin Rong. 2016. "word2vec Parameter Learning Explained." word2vec Parameter Learning Explained. https://arxiv.org/pdf/1411.2738.pdf.

"XLNet - transformers documentation." n.d. Hugging Face. Accessed April 14, 2022. https://huggingface.co/docs/transformers/model_doc/xlnet.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. “ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS.” ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS. https://arxiv.org/pdf/1909.11942.pdf.

15. Apèndix

15.1. Prova perplexitat, resultats plans

15.1.1. Perplexity 1


```
{
  "minAA":0.09430833836578383,
  "maxAA":0.7204792511017348,
  "varianceAA":0.06364283417079626,
  "meanAA":0.3715264050059517,
  "minAB":0.39036556317350946,
  "maxAB":0.759273383945246,
  "varianceAB":0.01709868371003545,
  "meanAB":0.5783810293950536,
  "minBB":0.2392005149529435,
  "maxBB":0.5093734222583071,
  "varianceBB":0.012059973916080612,
  "meanBB":0.3554
}
```


15.1.2. Perplexity 4


```
{
  "minAA":0.1368294092583718,
  "maxAA":0.5796438305489484,
  "varianceAA":0.033211937042941835,
  "meanAA":0.32024015616011126,
  "minAB":0.3829081494007976,
  "maxAB":0.8368230396170776,
  "varianceAB":0.021087211514857734,
  "meanAB":0.6491525421543182,
  "minBB":0.24843044098869393,
  "maxBB":0.5458634874649513,
  "varianceBB":0.011575319327561666,
  "meanBB":0.3853337473483368
}
```


15.1.3. Perplexity 9


```
{
  "minAA":0.1724155398215534,
  "maxAA":0.6200443047483907,
  "varianceAA":0.03216214251760882,
  "meanAA":0.3615472361342991,
  "minAB":0.36654903084473217,
```

```
  "maxAB":0.8188336538909798,
  "varianceAB":0.02140774488945103,
  "meanAB":0.6169891065528338,
  "minBB":0.1867075354969649,
  "maxBB":0.5684348398469187,
  "varianceBB":0.016711753480245676,
  "meanBB":0.3640233554727873
}
```


15.1.4. Perplexity 16


```
{
  "minAA":0.18201961217924179,
  "maxAA":0.661708238521128,
  "varianceAA":0.029516325177440964,
  "meanAA":0.39579647347998387,
  "minAB":0.310107496130478,
  "maxAB":0.806665515305678,
  "varianceAB":0.02710911518644612,
  "meanAB":0.566194887665158,
  "minBB":0.19669192015471684,
  "maxBB":0.6198015141049715,
  "varianceBB":0.022919832584960064,
  "meanBB":0.3986180098286651
}
```


15.1.5. Perplexity 25


```
{
  "minAA":0.1945727082810004,
  "maxAA":0.8105391965805999,
  "varianceAA":0.04697753725314311,
  "meanAA":0.4859723008453488,
  "minAB":0.19594909104359634,
  "maxAB":0.8091311185703121,
  "varianceAB":0.044087185897662674,
  "meanAB":0.48356962344485416,
  "minBB":0.21780245936073941,
  "maxBB":0.7885109068260142,
  "varianceBB":0.04202487513052321,
  "meanBB":0.48758278472824446
}
```


15.1.6. Perplexity 36


```
{
  "minAA":0.19583076485445233,
  "maxAA":0.7657111643632089,
  "varianceAA":0.04063269498449434,
  "meanAA":0.4528945281328966,
  "minAB":0.21370030736339096,
```

```
  "maxAB":0.7887863741492791,
  "varianceAB":0.0412159100276212,
  "meanAB":0.4990972922781819,
  "minBB":0.22138675104703553,
  "maxBB":0.8197571339338147,
  "varianceBB":0.04385412557981766,
  "meanBB":0.5093859423257596
}
```


15.1.7. Perplexity 49


```
{
  "minAA":0.1879793804016087,
  "maxAA":0.7830092765289269,
  "varianceAA":0.044321351471475874,
  "meanAA":0.47458512774508393,
  "minAB":0.2076566477418514,
  "maxAB":0.8010371705097277,
  "varianceAB":0.04291348219272279,
  "meanAB":0.4843732110634454,
  "minBB":0.19117210077346627,
  "maxBB":0.8080442382635,
  "varianceBB":0.045210349929620315,
  "meanBB":0.47863195679854836
}
```


15.1.8. Perplexity 64


```
{
  "minAA":0.21782740970969944,
  "maxAA":0.8081753435060325,
  "varianceAA":0.047443610822615556,
  "meanAA":0.49486129935784623,
  "minAB":0.23558185318211228,
  "maxAB":0.7815443896661783,
  "varianceAB":0.038013488282752805,
  "meanAB":0.48601250224984693,
  "minBB":0.22496008110868154,
  "maxBB":0.7974409600057084,
  "varianceBB":0.0424742938837929,
  "meanBB":0.49950451824211045
}
```


15.1.9. Perplexity 81


```
{
  "minAA":0.19769220899614065,
  "maxAA":0.783071224347608,
  "varianceAA":0.041930524149659906,
  "meanAA":0.470672047041827,
  "minAB":0.1943991948961187,
```

```
  "maxAB":0.8212146363532576,
  "varianceAB":0.04439074526178018,
  "meanAB":0.4931220378307903,
  "minBB":0.2089161898688548,
  "maxBB":0.7913283752857425,
  "varianceBB":0.040729823735415685,
  "meanBB":0.48729958761848824
}
```


15.1.10. Perplexity 100


```
{
  "minAA":0.22024702661772122,
  "maxAA":0.7981423171768612,
  "varianceAA":0.04440973748923628,
  "meanAA":0.4901627257816318,
  "minAB":0.25273442126686657,
  "maxAB":0.8266609887504063,
  "varianceAB":0.041780520193308635,
  "meanAB":0.537732232568259,
  "minBB":0.21262270787838775,
  "maxBB":0.8222795629860256,
  "varianceBB":0.04966301432733221,
  "meanBB":0.5092757980775177
}
```


15.1.11. Perplexity 121


```
{
  "minAA":0.19667255664962613,
  "maxAA":0.7899850714946166,
  "varianceAA":0.04443042215905913,
  "meanAA":0.48408573949723666,
  "minAB":0.20104205690856353,
  "maxAB":0.8142646169258694,
  "varianceAB":0.04611906121273913,
  "meanAB":0.4990817056426944,
  "minBB":0.22181731344922448,
  "maxBB":0.805525204374948,
  "varianceBB":0.043895214697824175,
  "meanBB":0.498608097798787
}
```


15.1.12. Perplexity 144


```
{
  "minAA":0.20455202179267923,
  "maxAA":0.772031660740178,
  "varianceAA":0.04128862503821105,
  "meanAA":0.4669185546645418,
  "minAB":0.2055904326209162,
```

```
  "maxAB":0.7952868023118452,
  "varianceAB":0.040797025913962046,
  "meanAB":0.47176592674604717,
  "minBB":0.22394644836016603,
  "maxBB":0.799856760543418,
  "varianceBB":0.042229498994777834,
  "meanBB":0.49039198090232
}
```


15.1.13. Perplexity 169


```
{
  "minAA":0.20339769361141,
  "maxAA":0.7966037501863388,
  "varianceAA":0.044477739759556156,
  "meanAA":0.49381086546321606,
  "minAB":0.22094617650978116,
  "maxAB":0.8102198150861555,
  "varianceAB":0.04388376749750685,
  "meanAB":0.5048676391117245,
  "minBB":0.19552887771055705,
  "maxBB":0.7967707000724228,
  "varianceBB":0.04708283315509309,
  "meanBB":0.4908606706526566
}
```


15.1.14. Perplexity 196


```
{
  "minAA":0.215101054119142,
  "maxAA":0.8205363969854768,
  "varianceAA":0.04808558930419542,
  "meanAA":0.5043860557123235,
  "minAB":0.21533504927232866,
  "maxAB":0.8261322544887835,
  "varianceAB":0.04505725672124686,
  "meanAB":0.5151088827608686,
  "minBB":0.21289027600180177,
  "maxBB":0.8199366871308081,
  "varianceBB":0.04861874560557671,
  "meanBB":0.5039131040436395
}
```

15.2. Script python notebook prova perplexitat

```
#!/usr/bin/env python
# coding: utf-8

# Aquest fitxer ens servirà per avaluar els resultats d'una execució de
perplexitats en general.
#

# Primer, carreguem tots els jsons que volem fer servir:
#

# In[2]:


path = "../results/execute_perplexity_test2021-07-12-18-40-58" #
directori on son tots els jsons


# In[3]:


import os
import smart_open
import ast
import pandas as pd
from scipy import stats
import numpy as np
import functools
import operator


# In[4]:


def process_json(path):
    with smart_open.open(path, 'r', encoding="utf-8") as f:
        content = ""
        for _, line in enumerate(f):
            content += line
```

```
        return [
            ast.literal_eval(content)["list"],
            ast.literal_eval(content)["method"],
            ast.literal_eval(content)["metric"],
            ast.literal_eval(content)["model"],
            ast.literal_eval(content)["context"],
            ast.literal_eval(content)["perplexity"]
        ]


# In[5]:


def merge_dicts(d1, d2):
    d = {}
    for k in d1:
        d[k] = d1[k] + d2[k]
    return d


# In[6]:


def transform_list_to_matrix(l):
    x = []
    for elem in l:
        y = []
        for unit in elem["vecs"]:
            y.append(float(unit["value"]))
        x.append(y)
    return x


# In[7]:


def traverse_matrix(i, imax, j, jmax, m):
    q = []

    while i < imax:
        k = j
        while k < jmax:
            q.append(m[i][j])
```

```
            k += 1
        i += 1
    return np.array(q)

def clean_class(q):
    q = q[q != 0]
    q = q if len(q) != 0 else [1]
    return q

def get_metrics_from_matrix(m):
    size = len(m)

    q1 = traverse_matrix(0, size//2, 0, size//2, m)
    q2 = traverse_matrix(size//2, size, 0, size//2, m)
    q4 = traverse_matrix(size//2, size, size//2, size, m)


    q1 = clean_class(q1)
    q2 = clean_class(q2)
    q4 = clean_class(q4)


    return {
        'minAA': [stats.describe(q1).minmax[0]],
        'maxAA': [stats.describe(q1).minmax[1]],
        'varianceAA': [stats.describe(q1).variance],
        'meanAA': [stats.describe(q1).mean],
        'minAB': [stats.describe(q2).minmax[0]],
        'maxAB': [stats.describe(q2).minmax[1]],
        'varianceAB': [stats.describe(q2).variance],
        'meanAB': [stats.describe(q2).mean],
        'minBB': [stats.describe(q4).minmax[0]],
        'maxBB': [stats.describe(q4).minmax[1]],
        'varianceBB': [stats.describe(q4).variance],
        'meanBB': [stats.describe(q4).mean]
    }


# In[8]:


def get_info_dict(file):
    [l, method, metric, model, context, perplexity] = process_json(file)
```

```
    m = transform_list_to_matrix(l)

    q = {
        'method': [method],
        'metric': [metric],
        'model': [model],
        'context': [context],
        'perplexity': [perplexity],
        'file': [file]
    }

    print(q)


    return {**get_metrics_from_matrix(m), **q}



# In[9]:



# test_file = os.path.join(path, os.listdir(path)[0],
os.listdir(os.path.join(path, os.listdir(path)[0]))[0])

# final = get_info_dict(test_file)

# print(final)

# https://datatofish.com/create-pandas-dataframe/
# merged = merge_dicts(final, final)
# df = pd.DataFrame(merged, columns=merged.keys())
# print(df)


# La idea és obtindre algo així de tots els fitxers:
# path, method, metric, model, context, minAA, maxAA, varianceAA,
meanAA, minAB, maxAB, varianceAB, meanAB, minBB, maxBB, varianceBB,
meanBB, NAs, perplexity
#
# on mean XY és la mitja del quadrant XY (per classes)
```

```
# NAs -> a vegades hi han overflows al càlcul d'alguna distància,
tindre-ho en compte

# In[ ]:




# In[10]:


# for txt in os.listdir(path):
#     print(txt)


# In[11]:


# path = "execute_perplexity_test2021-07-12-18-40-58"
f = []
for dr in os.listdir(path):
   for json in os.listdir(os.path.join(path, dr)):
       print("getting from " + json)
       d = get_info_dict(os.path.join(os.path.join(path, dr),json))
#         print(d)
       f.append(d)
merged = functools.reduce(lambda x, y: merge_dicts(x, y), f)

df = pd.DataFrame(merged, columns=merged.keys())
print(df)


# In[12]:


print(df.describe())


# In[37]:
```

```
# agrupar per perplexity
# fa grups : {[perplexity]: df}
# print()
grs = dict(tuple(df.groupby('perplexity')))


# fem mitjes de tots els paràmetres (maxAA, minAA, ...)
dic = {}
for k, v in grs.items():
   dic[k] = {}
   for c, vv in v.items():
       if c != 'method' and c != 'metric' and c != 'model' and c !=
'context' and c != 'perplexity' and c != 'file':
#             print(k + ":" + c + " -> " + str(v[c].mean()))
           dic[k][c] = v[c].mean()
print(dic)

import collections

dic = {int(k):v for k,v in dic.items()}
dic = collections.OrderedDict(sorted(dic.items()))


# In[56]:


import matplotlib.pyplot as plt
import numpy as np


maxaa = []
maxab = []
maxbb = []
for k, v in dic.items():
   maxaa.append(dic[k]['maxAA'])
   maxab.append(dic[k]['maxAB'])
   maxbb.append(dic[k]['maxBB'])


np.arange(3)
width = 0.2

fig, axs = plt.subplots(figsize=(10, 5))
```

```
plt.bar(x-0.2, maxaa, width, color='cyan')
plt.bar(x, maxab, width, color='orange')
plt.bar(x+0.2, maxbb, width, color='green')
axs.legend(["maxAA", "maxAB", "maxBB"])
axs.set_title("Maximum values comparation")
axs.set(xlabel="perplexity")

plt.sca(axs)
plt.xticks(x, [k for k, _ in dic.items()])
plt.show()


# In[52]:


minaa = []
minab = []
minbb = []
for k, v in dic.items():
   minaa.append(dic[k]['minAA'])
   minab.append(dic[k]['minAB'])
   minbb.append(dic[k]['minBB'])


np.arange(3)
width = 0.2

fig, axs = plt.subplots(figsize=(10,5))
plt.bar(x-0.2, minaa, width, color='cyan')
plt.bar(x, minab, width, color='orange')
plt.bar(x+0.2, minbb, width, color='green')
axs.legend(["minAA", "minAB", "minBB"])
axs.set_title("Minimum values comparation")
axs.set(xlabel="perplexity")

plt.sca(axs)
plt.xticks(x, [k for k, _ in dic.items()])
plt.show()


# In[57]:
```

```
varaa = []
varab = []
varbb = []
for k, v in dic.items():
   varaa.append(dic[k]['varianceAA'])
   varab.append(dic[k]['varianceAB'])
   varbb.append(dic[k]['varianceBB'])


np.arange(3)
width = 0.2

fig, axs = plt.subplots(figsize=(10,5))
plt.bar(x-0.2, varaa, width, color='cyan')
plt.bar(x, varab, width, color='orange')
plt.bar(x+0.2, varbb, width, color='green')
axs.legend(["varianceAA", "varianceAB", "varianceBB"])
axs.set_title("Variance values comparation")
axs.set(xlabel="perplexity")

plt.sca(axs)
plt.xticks(x, [k for k, _ in dic.items()])
plt.show()


# In[58]:


meanaa = []
meanab = []
meanbb = []
for k, v in dic.items():
   meanaa.append(dic[k]['meanAA'])
   meanab.append(dic[k]['meanAB'])
   meanbb.append(dic[k]['meanBB'])


np.arange(3)
width = 0.2

fig, axs = plt.subplots(figsize=(10,5))
plt.bar(x-0.2, meanaa, width, color='cyan')
plt.bar(x, meanab, width, color='orange')
```

```
plt.bar(x+0.2, meanbb, width, color='green')
axs.legend(["meanAA", "meanAB", "meanBB"])
axs.set_title("Mean values comparation")
axs.set(xlabel="perplexity")

plt.sca(axs)
plt.xticks(x, [k for k, _ in dic.items()])
plt.show()
```

15.3. Exemple dataframe prova perplexitat

	minAA	maxAA	varianceAA	meanAA	minAB	maxAB	varianceAB	\
0	0.008505	0.999869	0.138044	0.643756	0.113829	0.998189	0.073679	
1	0.021931	0.978135	0.142819	0.369669	0.221670	0.931963	0.072402	
2	0.230737	0.985867	0.064163	0.599404	0.256010	0.931382	0.040558	
3	0.222272	0.734831	0.024097	0.403193	0.197361	0.778813	0.037200	
4	0.153168	0.910357	0.061974	0.464886	0.159686	0.913218	0.052386	
..	...	...	...	...	...	...	...	
835	0.262410	0.984074	0.045247	0.580019	0.225916	0.978490	0.059047	
836	0.226340	0.801457	0.032458	0.387516	0.212506	0.794357	0.028922	
837	0.236348	0.698739	0.027982	0.441056	0.224569	0.821722	0.036953	
838	0.241974	0.922303	0.044151	0.557891	0.248102	0.871436	0.048590	
839	0.206225	0.849911	0.057745	0.473544	0.153102	0.883612	0.043909	

	meanAB	minBB	maxBB	varianceBB	meanBB	method	metric	\
0	0.588977	0.077971	0.992055	0.137159	0.655787	NXNROWS	COSINE	
1	0.559731	0.000727	0.999739	0.135606	0.603207	NXNROWS	COSINE	
2	0.607302	0.243157	0.877755	0.042164	0.579622	NXNROWS	MINKOWSKI_2	
3	0.469132	0.194885	0.726272	0.035775	0.465645	NXNROWS	MINKOWSKI_3	
4	0.556644	0.196929	0.893749	0.041290	0.474266	NXNROWS	HAVERSINE	
..	...	...	...	...	...	...	...	
835	0.635447	0.262886	1.000000	0.062963	0.628189	NXNROWS	MINKOWSKI_2	
836	0.557586	0.233210	0.594776	0.015093	0.368956	NXNROWS	MINKOWSKI_2	
837	0.590591	0.205852	0.889685	0.047033	0.525491	NXNROWS	MANHATTAN	
838	0.558935	0.213814	0.711492	0.036524	0.442968	NXNROWS	EUCLIDEAN	
839	0.552855	0.129950	0.888802	0.057916	0.569493	NXNROWS	HAVERSINE	

	model	context	perplexity	\
0	DOC2VEC	SCIENTIFIC	64	
1	DOC2VEC	WIKI	64	

```
2   WORD2VEC  SCIENTIFIC       64
3   WORD2VEC  SCIENTIFIC       64
4    DOC2VEC       WIKI       64
..       ...        ...       ...
835  WORD2VEC      BLOGS       81
836  WORD2VEC       WIKI       81
837   DOC2VEC       WIKI       81
838   DOC2VEC  SCIENTIFIC       81
839   DOC2VEC       WIKI       81

                                  file
0    ../results/execute_perplexity_test2021-07-12-1...
1    ../results/execute_perplexity_test2021-07-12-1...
2    ../results/execute_perplexity_test2021-07-12-1...
3    ../results/execute_perplexity_test2021-07-12-1...
4    ../results/execute_perplexity_test2021-07-12-1...
..                                   ...
835  ../results/execute_perplexity_test2021-07-12-1...
836  ../results/execute_perplexity_test2021-07-12-1...
837  ../results/execute_perplexity_test2021-07-12-1...
838  ../results/execute_perplexity_test2021-07-12-1...
839  ../results/execute_perplexity_test2021-07-12-1...

[840 rows x 18 columns]
```

15.4. Script perplexitat 4 discriminació per coeficients

```
def printCoefs(df, title):
      avgMaxABAA = df["maxAB"].mean() - df["maxAA"].mean()
      avgMaxABBB = df["maxAB"].mean() - df["maxBB"].mean()
      avgMinABAA = df["minAB"].mean() - df["minAA"].mean()
      avgMinABBB = df["minAB"].mean() - df["minBB"].mean()
      avgMeanABAA = df["meanAB"].mean() - df["meanAA"].mean()
      avgMeanABBB = df["meanAB"].mean() - df["meanBB"].mean()
      avgVarABAA = df["varianceAB"].mean() - df["varianceAA"].mean()
      avgVarABBB = df["varianceAB"].mean() - df["varianceBB"].mean()

      import matplotlib.pyplot as plt
      import numpy as np

      x = np.arange(4)
      width = 0.40

      y1 = [avgMaxABAA, avgMinABAA, avgMeanABAA, avgVarABAA]
```

```
      y2 = [avgMaxABBB, avgMinABBB, avgMeanABBB, avgVarABBB]

      plt.bar(x-0.2, y1, width)
      plt.bar(x+0.2, y2, width)
      plt.xticks(x, ["AvgMax", "AvgMin", "AvgMean", "AvgVar"])
      plt.legend(["ABAA", "ABBB"])
      plt.title(title)
```

15.5. Script perplextat 4 discriminació per grups

```
# type data can be "max"|"min"|"mean"|"variance"
def printMetrics(df, groupbycolumn, typedata, title):

      # agrupar per metric
      # fa grups : {[metric]: df}
      grs = dict(tuple(df.groupby(groupbycolumn)))

      # fem mitjes de tots els paràmetres (maxAA, minAA, ...)
      dic = {}
      for k, v in grs.items():
      dic[k] = {}
      for c, vv in v.items():
            if c != 'method' and c != 'metric' and c != 'model' and c !=
'context' and c != 'perplexity' and c != 'file':
      #           print(k + ":" + c + " -> " + str(v[c].mean()))
                  dic[k][c] = v[c].mean()

      # 2x2 subplots, un per max, altre min, altre mean i finalment
variance

      ndic = {}
      for k, v in dic.items():

      avgABAA = None
      avgABBB = None

      if typedata == "max":
            avgABAA = v["maxAB"].mean() - v["maxAA"].mean()
            avgABBB = v["maxAB"].mean() - v["maxBB"].mean()
      elif typedata == "min":
            avgABAA = v["minAB"].mean() - v["minAA"].mean()
            avgABBB = v["minAB"].mean() - v["minBB"].mean()
      elif typedata == "mean":
            avgABAA = v["meanAB"].mean() - v["meanAA"].mean()
```

```
            avgABBB = v["meanAB"].mean() - v["meanBB"].mean()
      else:
            avgABAA = v["varianceAB"].mean() - v["varianceAA"].mean()
            avgABBB = v["varianceAB"].mean() - v["varianceBB"].mean()


      ndic[k] = {
            'avgABAA': avgABAA,
            'avgABBB': avgABBB
      }


      import matplotlib.pyplot as plt
      import numpy as np
      abaa = []
      abbb = []

      for k, v in ndic.items():
      abaa.append(ndic[k]['avgABAA'])
      abbb.append(ndic[k]['avgABBB'])


      width = 0.3
      x = np.arange(len(abaa))


      fig, axs = plt.subplots(figsize=(10,5))
      plt.bar(x-(width/2), abaa, width, color='cyan')
#     plt.bar(x, minab, width, color='orange')
      plt.bar(x+(width/2), abbb, width, color='green')
      axs.legend(["avgABAA", "avgABBB"])
      axs.set_title(title)
#     axs.set(xlabel="perplexity")

      plt.sca(axs)
      plt.xticks(x, [k for k, _ in ndic.items()], rotation=60)
      plt.show()
```

15.6. Script perplexitat 4 discriminació per grups isolats

```
# type data can be "max"|"min"|"mean"|"variance"
def printMetricsIsolated(df, groupbycolumn, typedata, title):
```

```
      # agrupar per metric
      # fa grups : {[metric]: df}
      # print()
      grs = dict(tuple(df.groupby(groupbycolumn)))

      # fem mitjes de tots els paràmetres (maxAA, minAA, ...)
      dic = {}
      for k, v in grs.items():
      dic[k] = {}
      for c, vv in v.items():
            if c != 'method' and c != 'metric' and c != 'model' and c !=
'context' and c != 'perplexity' and c != 'file':
      #           print(k + ":" + c + " -> " + str(v[c].mean()))
                  dic[k][c] = v[c].mean()

      # 2x2 subplots, un per max, altre min, altre mean i finalment
variance

      ndic = {}
      for k, v in dic.items():

      avgAA = None
      avgAB = None
      avgBB = None

      if typedata == "max":
            avgAA = v["maxAA"].mean()
            avgAB = v["maxAB"].mean()
            avgBB = v["maxBB"].mean()
      elif typedata == "min":
            avgAA = v["minAA"].mean()
            avgAB = v["minAB"].mean()
            avgBB = v["minBB"].mean()
      elif typedata == "mean":
            avgAA = v["meanAA"].mean()
            avgAB = v["meanAB"].mean()
            avgBB = v["meanBB"].mean()
      else:
            avgAA = v["varianceAA"].mean()
            avgAB = v["varianceAB"].mean()
            avgBB = v["varianceBB"].mean()


      ndic[k] = {
            'avgAA': avgAA,
```

```
            'avgAB': avgAB,
            'avgBB': avgBB
        }


        import matplotlib.pyplot as plt
        import numpy as np
        aa = []
        ab = []
        bb = []

        for k, v in ndic.items():
        aa.append(ndic[k]['avgAA'])
        ab.append(ndic[k]['avgAB'])
        bb.append(ndic[k]['avgBB'])


        width = 0.2
        x = np.arange(len(aa))


        fig, axs = plt.subplots(figsize=(10,5))
        plt.bar(x-(width), aa, width, color='cyan')
        plt.bar(x, ab, width, color='orange')
        plt.bar(x+(width), bb, width, color='green')
        axs.legend(["avgSC", "avgMixed", "avgBLOGS"])
        axs.set_title(title)
#       axs.set(xlabel="perplexity")

        plt.sca(axs)
        plt.xticks(x, [k for k, _ in ndic.items()], rotation=60)
        plt.show()
```