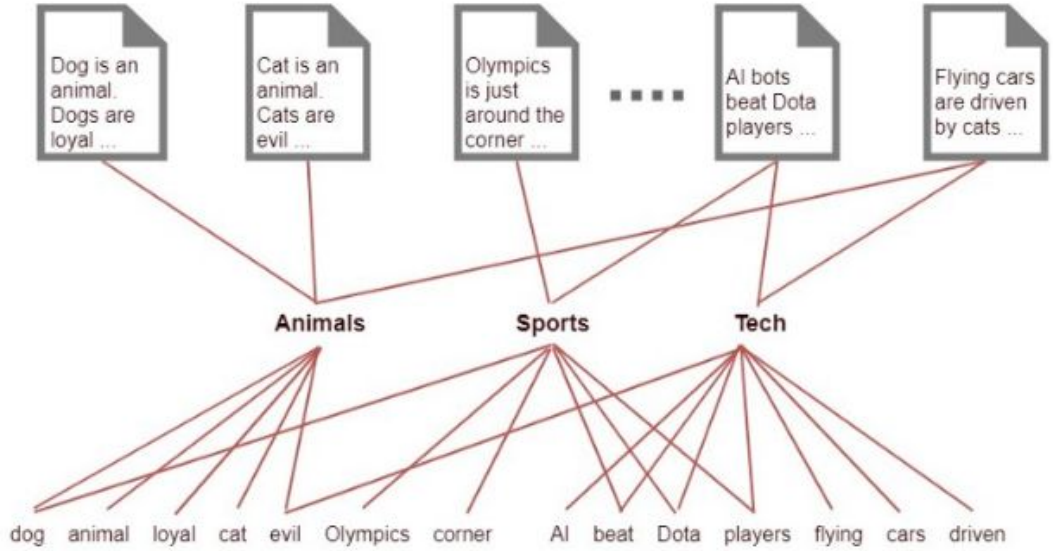
Proyecto Final
Entrega Final

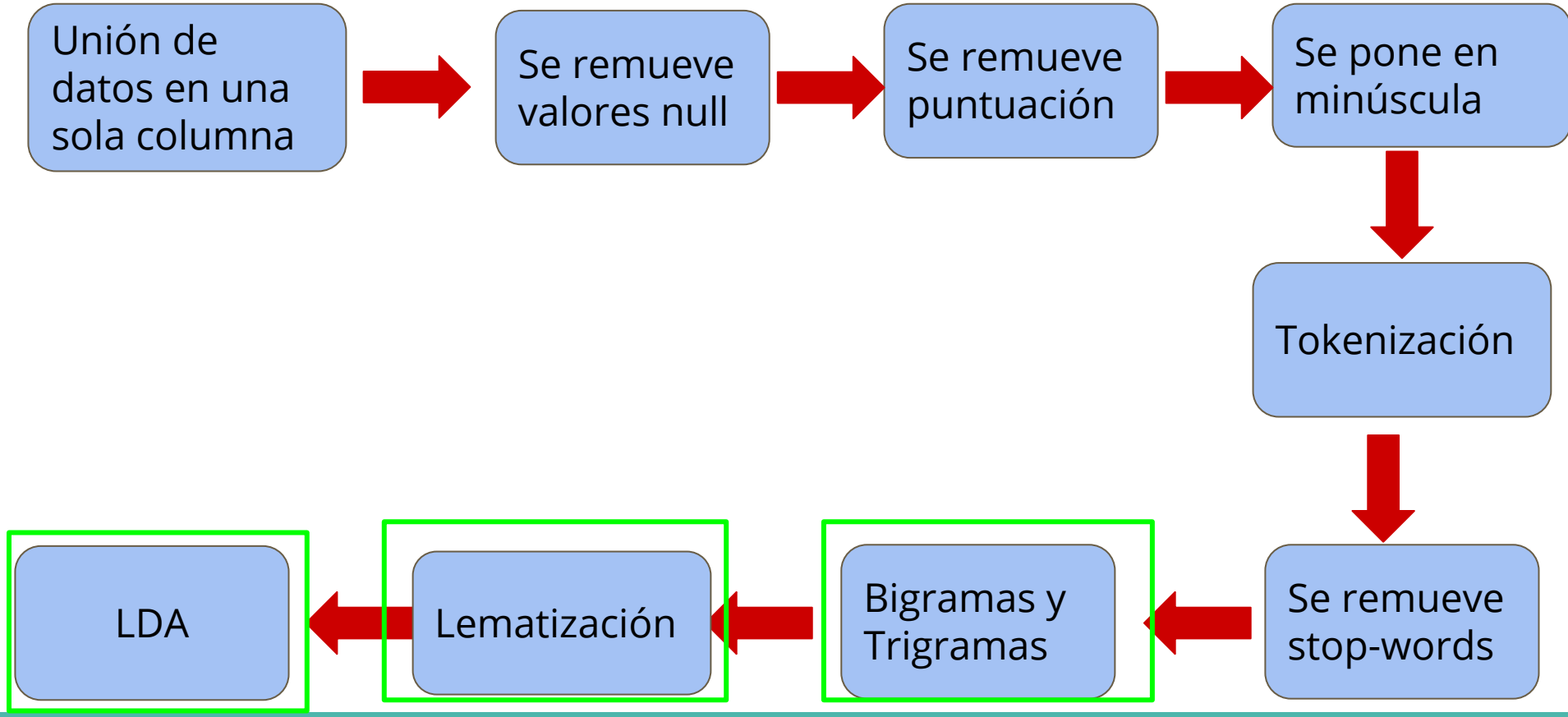
Sandra Niño
Johnatan Garzón

LDA (Latent Dirichlet Allocation)

Cada documento puede ser descrito por una distribución de “topics” y cada topic puede ser descrito por una distribución de palabras.



Procedimiento



Remover Stop words

Remover Stop words ¶

Removemos las palabras que son conocidas como "Stop words": lo, la, con, antes, sin, de, entre otras, de los datos tokenizados.

```
stop_words_exceptions=["no"]
def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc)) if (word not in stop_words) or (word in stop_words_exception

data_tokenized_nostops= remove_stopwords(data_tokenized)
```

{'demas_personas', 'buena_convivencia', 'sentirse_tranquilo', 'cada_persona', 'conmigo_misma', 'poder_compartir', '
respeto_tolerancia', 'seres_queridos', 'gracias_dios', 'buenas_personas', 'buena_comunicacion', 'ninos_jugando', 'b
ien_conmigo', 'equidad_social', 'conmigo_mismo', 'mas_alla', 'sigo_mismo', 'poder_convivir', 'entorno_protector', '
ninos_puedan', 'derechos_humanos', 'iniciativas_comunitarias', 'escuchar_musica', 'buenas_acciones', 'cultura_ciuda
dana', 'puedan_salir', 'mas_importante', 'sana_convivencia', 'poder_salir', 'vivir_tranquilo', 'buenos_valores', 'n
o_discriminacion', 'ser_solidario', 'cualquier_lugar', 'personas_puedan', 'ser_libre', 'ambiente_sano', 'ninos_nina
s', 'tener_buena', 'mismos_derechos', 'ser_humano', 'asi_mismo', 'migo_mismo', 'poder_respirar', 'libre_feliz', 're
speto_hacia', 'libre_expresion', 'buena_relacion', 'convivencia_pacifica', 'derechos_sociales', 'si_generamos', 'co
nsigo_mismo', 'justicia_social', 'medio_ambiente', 'seres_vivos', 'fronteras_pasar', 'aprender_convivir', 'paz_', '
ser_tolerante', 'no_violencia', 'trabajo_empleo', 'seres_humanos', 'recursos_naturales', 'hacer_actividades', 'juga
r_futbol'}
tamano lista de bigramas sin stopwords: 1124
tamano lista de bigramas sin stopwords unicos: 65

{'libre_feliz', 'misma_comunidad', 'ser_humano', 'economica_social', 'derechos_sociales', 'mas_alla', 'ser_tolerantes
', 'poder_compartir', 'buenas_personas', 'union_familiar', 'ser_libre', 'migo_mismo', 'respeto_hacia', 'sana_conviven
cia', 'entorno_protector', 'podamos_convivir', 'asi_mismo', 'tolerancia_respeto', 'vida_espiritual', 'poder_hacer', '
cultura_ciudadana', 'gracias_dios', 'no_pelear', 'poder_disfrutar', 'estan_peleando', 'recursos_naturales', 'respeto_
tolerancia', 'hacen_parte', 'personas_puedan', 'cada_persona', 'siempre_pensando', 'demas_personas', 'conmigo_misma',
'paz_', 'familia_compartir', 'sigo_mismo', 'mas_importante', 'no_discriminacion_derechos_sociales', 'ninos_ninas', 'd
erechos_fundamentales', 'ala_vida', 'servir_ala', 'cada_vez', 'fronteras_pasar', 'derechos_humanos', 'dia_dia', 'no_d
iscriminacion', 'no_haber', 'puedan_salir', 'ninos_jugando', 'no_violencia_derechos_sociales', 'no_violencia', 'seres
_vivos', 'bien_conmigo', 'conceptos_opiniones', 'buena_relacion', 'hace_feliz', 'seres_queridos', 'poder_respirar', '
ser_solidario', 'ninos_puedan', 'buena_convivencia', 'conmigo_mismo', 'aprender_vivir', 'confianza_interpersonal', 'b
uenas_acciones', 'no_discriminacion_no_violencia', 'escuchar_musica', 'ser_amable', 'sentirse_tranquilo', 'vivir_tran
quilo', 'buena_comunicacion', 'mismos_derechos', 'ser_feliz', 'consigo_mismo', 'poder_dialogar', 'ambiente_sano', 'br
indar_amor', 'seres_humanos', 'tener_buena', 'jugar_futbol', 'cualquier_lugar', 'si_generamos', 'aprender_convivir',
'trabajo_empleo', 'buenos_valores', 'iniciativas_comunitarias', 'libre_expresion', 'poder_convivir', 'poder_jugar', '
participacion_ciudadana', 'medio_ambiente', 'poder_salir', 'tener_buena_convivencia', 'tanta_violencia', 'no_aparece
', 'salud_trabajo', 'justicia_social', 'deporte_arte', 'escucho_musica', 'buen_trato', 'hacer_actividades', 'conviven
cia_pacifica', 'alegria_felicidad', 'ser_tolerante', 'equidad_social'}
tamano lista de trigramas sin stopwords: 1385
tamano lista de trigramas sin stopwords unicos: 106

Lematización

```
allowed_postags=['ADJ', 'VERB', 'ADV']
stopwordsToken=["demas","tener","mismo", "poder","cada","tambien", "hacer"]
unifiedWord={"respetar":"respeto", "violencia":"no_violencia","musicar":"musica","guerra":"no_guerra"}
data_list=[]

for row in df.Narrativa_bigrams:
    token_list=""
    text=nlp(row)
    for token in text:
        if (token.pos_ in allowed_postags) and (token.lemma_ not in stopwordsToken) and (token.text not in stopwordsTo
            if token.lemma_ in unifiedWord:
                token_list+=unifiedWord[token.lemma_]+" "
            else :
                token_list+=token.lemma_+" "

        elif (token.is_stop is not True) and (token.lemma_ not in stopwordsToken) and (token.text not in stopwordsToke
            if token.text in unifiedWord:
                token_list+=unifiedWord[token.text]+" "
            else :
                token_list+=token.text+" "
    data_list.append(token_list.rstrip())
```

Resultado de limpieza de datos

LDA

Se crea una matriz Documento-Palabra como input para LDA

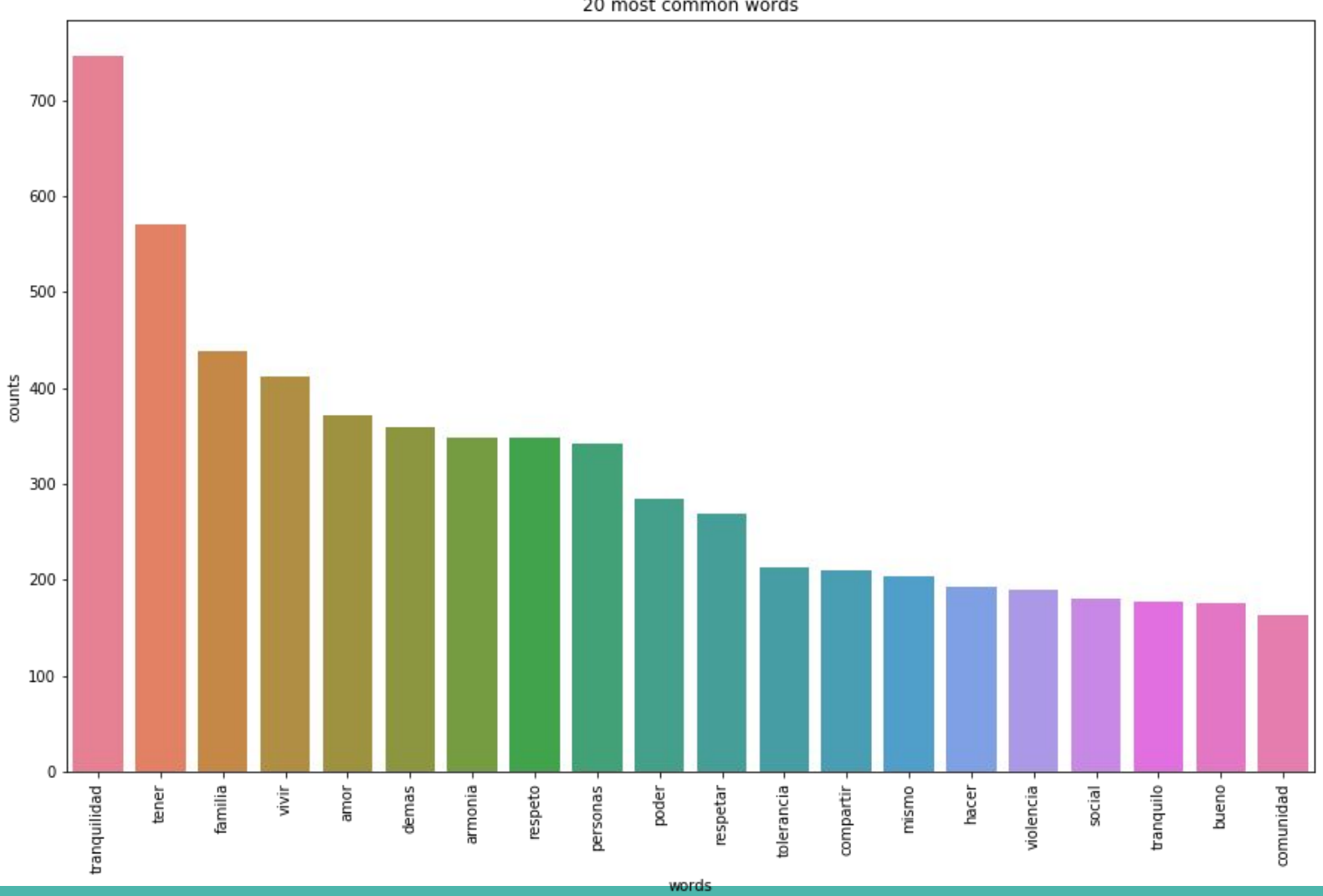
```
vectorizer = CountVectorizer(analyzer='word',
                             min_df=10,                          # minimo numero de ocurrencias
                             token_pattern='[a-z\_A-Z]{3,}',  # num chars > 3, que contenga bigramas (palabra_palabra)
                            )

data_vectorized = vectorizer.fit_transform(df['Narrativa_lemmatized'])
```

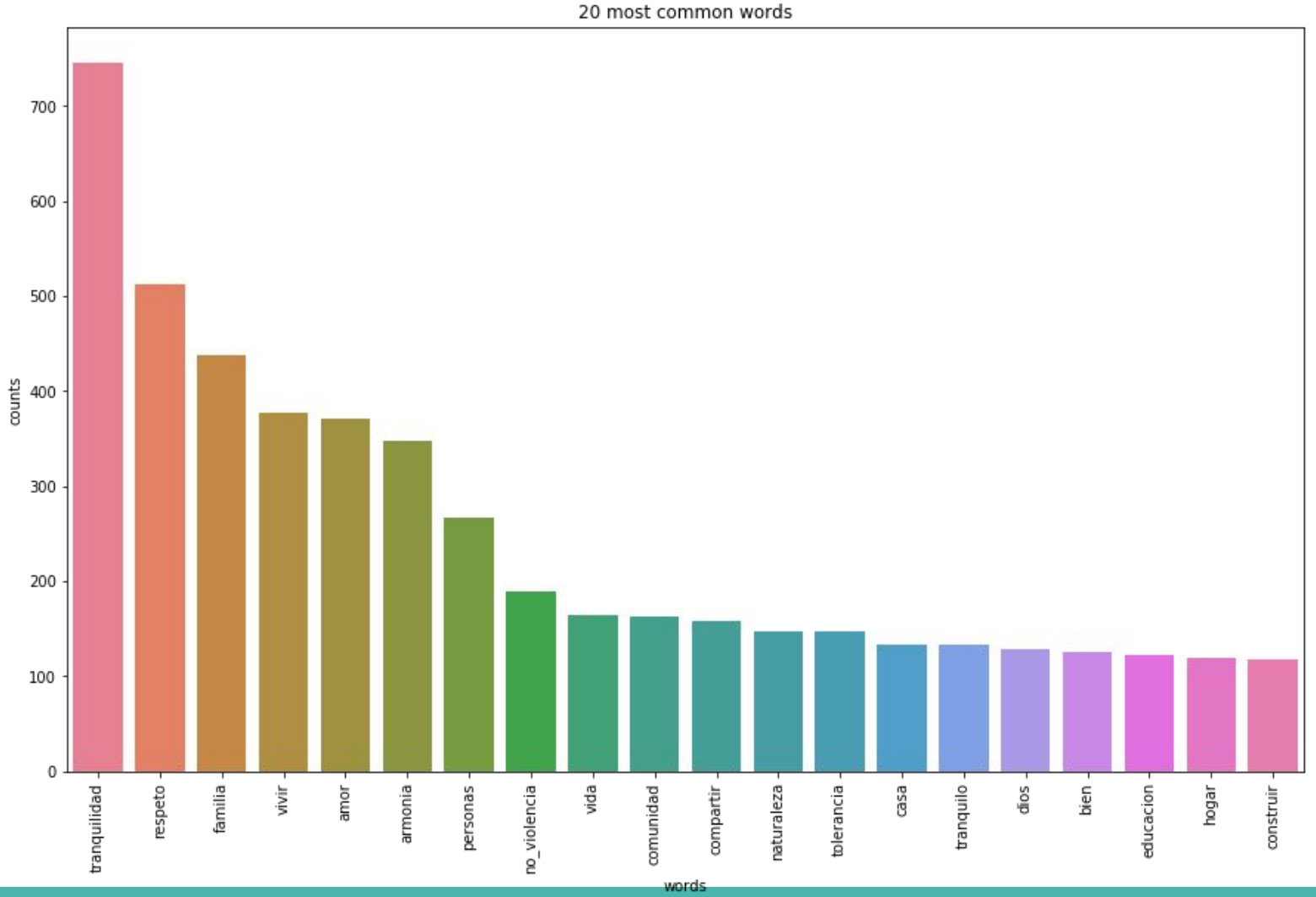
```
df.Narrativa_lemmatized[0]
```

```
'tranquilidad convivencia casa educacion colectividad respeto tranquilidad acceso educacion humanizacion espacio siem
pre camino'
```

```
print(data_vectorized)
```

```
  (0, 393)      1
  (0, 168)      1
  (0, 2)        1
  (0, 360)      1
  (0, 150)      2
  (0, 67)       1
  (0, 100)      1
  (0, 426)      2
  (1, 183)      1
```


20 most common words
counts
0
100
200
300
400
500
600
700
tranquilidad
tener
familia
vivir
amor
demas
armonia
respeto
personas
poder
respetar
tolerancia
compartir
mismo
hacer
violencia
social
tranquilo
bueno
comunidad
words


20 most common words
counts
0
100
200
300
400
500
600
700
tranquilidad
respeto
familia
vivir
amor
armonia
personas
no_violencia
vida
comunidad
compartir
naturaleza
tolerancia
casa
tranquilo
dios
bien
educacion
hogar
construir
words

Encontrar mejor LDA

Encontrar el mejor LDA

Realizaremos una búsqueda de parámetros para encontrar el mejor LDA. Para esto se tendrá en cuenta la variación de 2 parámetros:

- n_components, el cual es el número de topicos (grupos) que queremos formar.
- learning_decay, el cual controla la razón de aprendizaje.

Grid Search

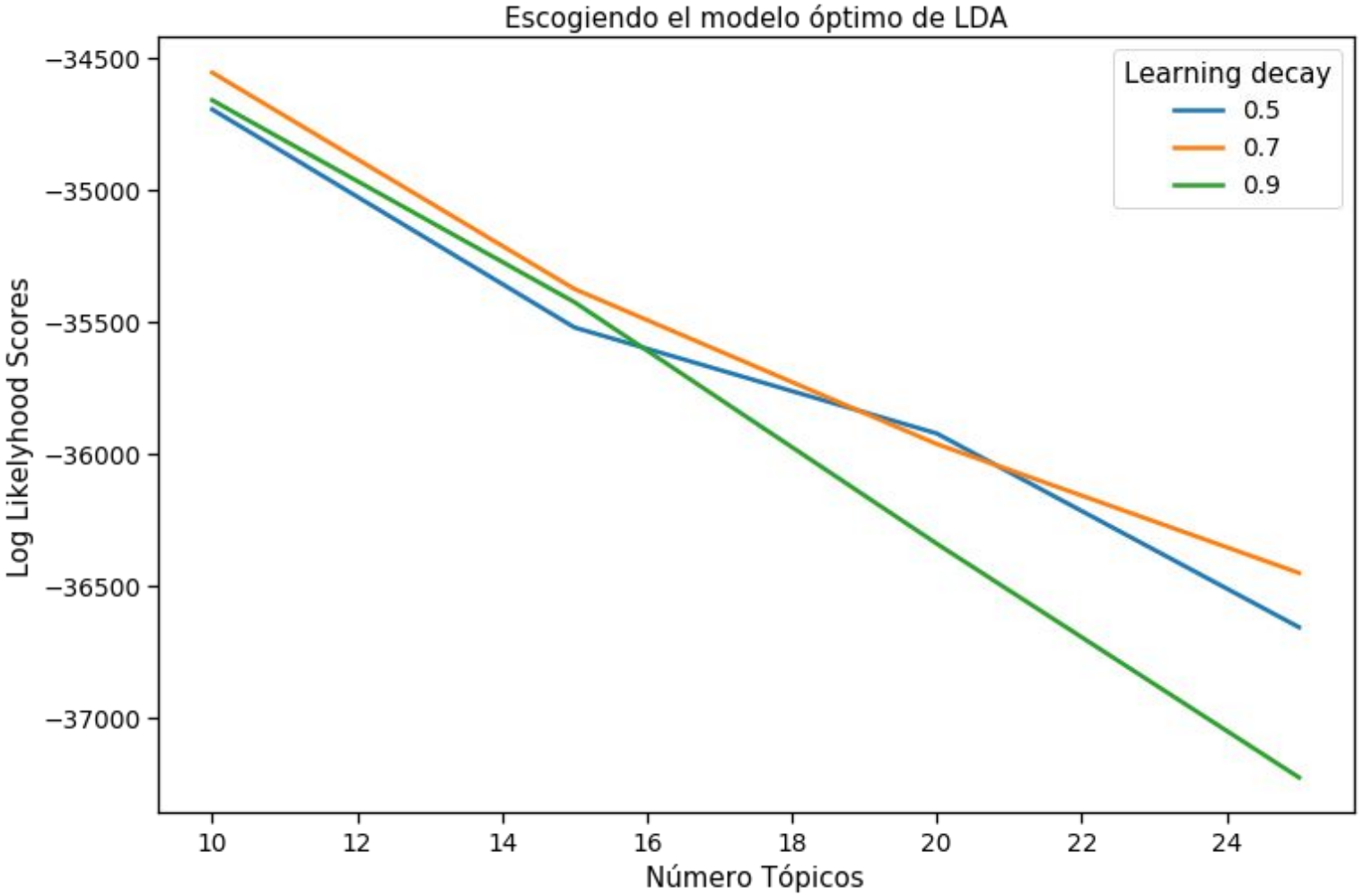
Este proceso de búsqueda de parámetros consume bastante tiempo, puesto que construye múltiples modelos de LDA para todas las posibles combinaciones de dichos parámetros en el diccionario **param_grid** (cross-validated grid-search).

```
# Se definen los parámetros de búsqueda
search_params = {'n_components': [10, 15, 20, 25], 'learning_decay': [.5, .7, .9]}

# Se inicializa el modelo
lda = LatentDirichletAllocation()

# Se inicializa la clase GridSearchCV
model = GridSearchCV(lda, param_grid=search_params)

# Se realiza Grid Search
model.fit(data_vectorized)
```

Escogencia del mejor modelo

```
# Best Model
lda_model = model.best_estimator_

# Model Parameters
print("Mejores parámetros del modelo: ", model.best_params_)

# Log Likelihood Score
print("Mejor Log Likelihood Score: ", model.best_score_)

# Perplexity
print("Perplexity del modelo: ", lda_model.perplexity(data_vectorized))
```

```
Mejores parámetros del modelo:  {'learning_decay': 0.7, 'n_components': 10, 'random_state': 100}
Mejor Log Likelihood Score:  -34554.234643328025
Perplexity del modelo:  358.6854330045643
```


Escogiendo el modelo óptimo de LDA
Learning decay
0.5
0.7
0.9
Log Likelyhood Scores
−34500
−35000
−35500
−36000
−36500
−37000
10
12
14
16
18
20
22
24
Número Tópicos

Tópico dominante en cada documento

	Topic0	Topic1	Topic2	Topic3	Topic4	Topic5	Topic6	Topic7	Topic8	Topic9	dominant_topic
Doc0	0.01	0.01	**0.44**	0.01	**0.25**	0.01	0.01	0.01	**0.25**	0.01	**2**
Doc1	0.01	0.01	0.08	**0.81**	0.01	0.01	0.01	0.01	0.01	0.07	**3**
Doc2	0.01	0.01	0.01	0.01	**0.23**	0.01	**0.6**	0.01	0.01	0.1	**6**
Doc3	**0.25**	0.01	0.01	0.01	**0.46**	0.01	0.01	0.01	0.01	**0.21**	**4**
Doc4	**0.33**	0.02	**0.26**	0.02	0.02	0.02	**0.27**	0.02	0.02	0.02	0
Doc5	**0.29**	**0.31**	0.01	0.01	**0.18**	**0.15**	0.01	0.01	0.01	0.01	**1**
Doc6	0.01	0.01	**0.61**	0.01	0.01	0.01	**0.29**	0.01	0.01	0.01	**2**
Doc7	0.03	0.03	0.03	0.03	**0.41**	0.03	**0.39**	0.03	0.03	0.03	**4**
Doc8	**0.19**	0.01	0.01	0.01	0.01	0.01	**0.67**	0.08	0.01	0.01	**6**
Doc9	0.01	0.01	**0.39**	0.01	0.01	0.01	**0.51**	0.01	0.01	0.01	**6**

Distribución de los tópicos en los documentos

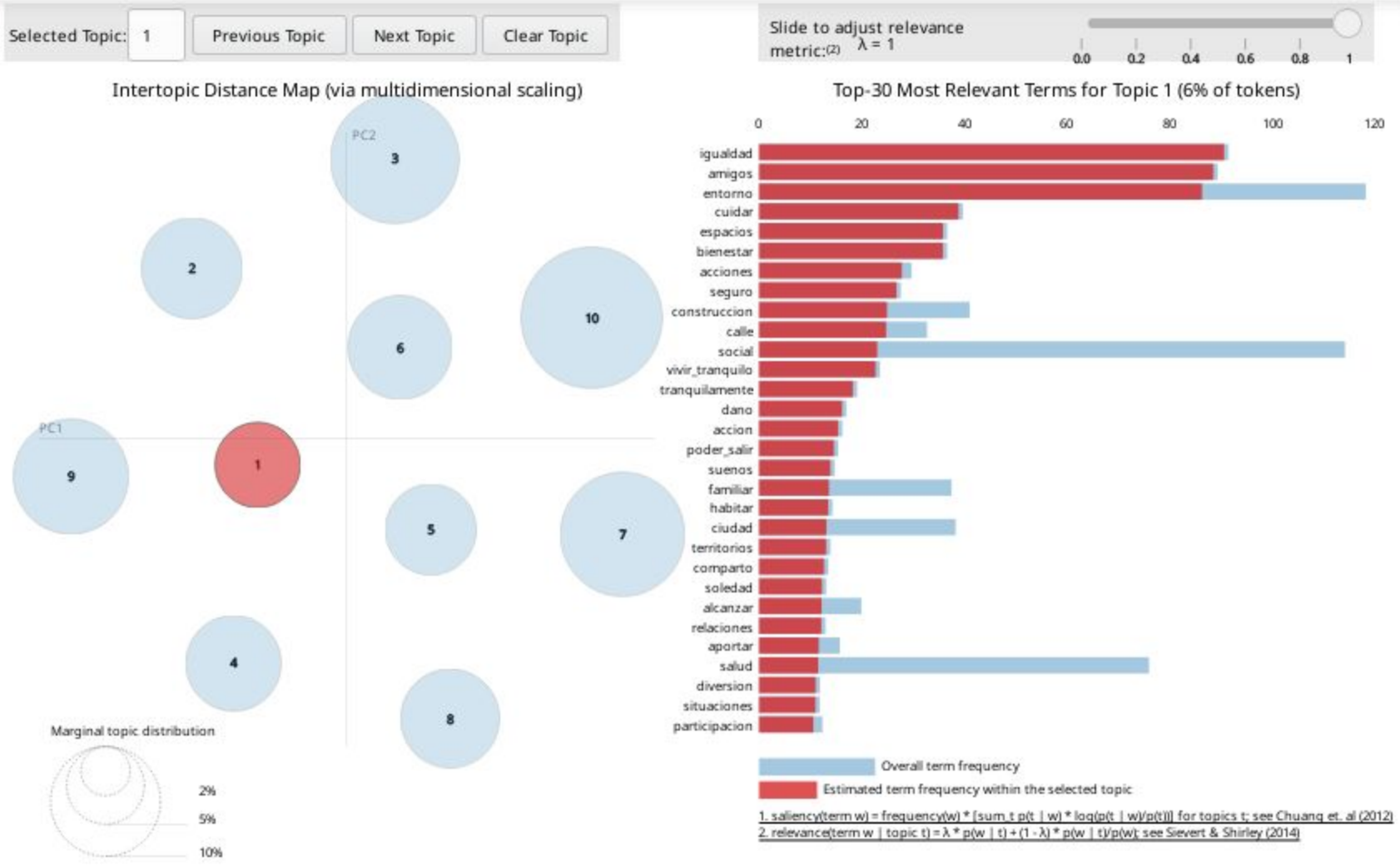
	Topic Num	Num Documents
0	9	474
1	4	378
2	6	368
3	2	363
4	8	273
5	3	229
6	1	221
7	5	214
8	0	213
9	7	201

10 palabras de cada tópico

```
Topic #0:
igualdad amigos entorno cuidar espacios bienestar acciones seguro construccion calle

Topic #1:
respeto personas demas_personas diferencias contar respeto_hacia tratar raza gente ambiente

Topic #2:
tranquilo sociedad no_violencia vida mejor educacion oportunidades social saber siempre

Topic #3:
armonia felicidad generar hijos corazon vida sana_convivencia alegria ser_humano barrio

Topic #4:
tranquilidad respeto solidaridad confianza personal tolerancia reflejar valores hogar dialogo

Topic #5:
construir mundo libertad respeto armonia lograr perdonar bueno derechos territorio

Topic #6:
amor respeto tolerancia pensar conflictos personas seguridad diferente persona respeto_tolerancia

Topic #7:
dios solo entender existir musica disfrutar escuchar projimo conflicto encuentro

Topic #8:
naturaleza casa sentir convivencia problemas representar llegar conciencia forma tener_buena

Topic #9:
familia vivir comunidad personas armonia compartir bien interior union bueno
```

pyLDAvis

Out[17]:



Se exporta el modelo para usarlo en la aplicación

Exportamos el modelo

Exportamos el modelo LDA para poder predecir futuros textos y clasificarlos en una categoria correspondiente.

```
from sklearn.externals import joblib

joblib.dump(lda_model,'modelo_entrenado.pkl')
```

```
['modelo_entrenado.pkl']
```

Exportamos el modelo Vectorizer para crear nuevas matrices de dispersión acorde a los datos entrenados.

```
joblib.dump(vectorizer,'modelo_vectorizer.pkl')
```

```
['modelo_vectorizer.pkl']
```

Importamos los modelos generados anteriormente

```
lda_model=joblib.load('modelo_entrenado.pkl')
```

```
vectorizer = joblib.load('modelo_vectorizer.pkl')
```

Procesamiento de un nuevo texto

Evaluaremos un nuevo texto, pero antes tenemos que hacer una respectiva limpieza.

```
text= "Para mi la paz es tener tranquilidad en mi hogar con mi familia, mis amigas y mi perro"
```

```
text= re.sub('[,\.\'\"!\)(?0-9]', '', text).lower().strip()
```

```
def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc= True))  # deacc=True removes punctuations

data = [text]
data_tokenized = list(sent_to_words(data))
```

```
stop_words_exceptions=["no"]
def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc)) if (word not in stop_words) or (word in stop_words_exceptions)]


data_tokenized_nostops= remove_stopwords(data_tokenized)
```

```
# Build the bigram and trigram models
bigram = gensim.models.Phrases(data_tokenized, min_count=7, threshold=5) # higher threshold fewer phrases.
```

```
4]: text_processed=vectorizer.transform(data_list)
    print(text_processed)
```

```
  (0, 181)      1
  (0, 214)      1
  (0, 426)      1
```

```
5]: topic_probability_scores = lda_model.transform(text_processed)
```

```
6]: print(topic_probability_scores)
```

```
[[0.0250002  0.025      0.02500212 0.02500561 0.51681747 0.025
  0.02500464 0.025      0.02500153 0.28316842]]
```

```
7]: np.argmax(topic_probability_scores)
```

```
7]: 4
```

```
8]: topic = df_topic_keywords.iloc[np.argmax(topic_probability_scores), :].values.tolist()
```

```
9]: print(', '.join(topic))
```

```
tranquilidad, respeto, solidaridad, confianza, personal, tolerancia, reflejar, valores, hogar, dialogo, construccion, uni
dad, medio_ambiente, adultos, generar, comun, interpersonal, practicar, normas, seres_humanos, bien, educacion, permitir,
caminar, comunitario, equidad, diferencia, diversidad, economica, derechos_sociales
```