UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

# DISEÑO Y CONSTRUCCIÓN DE UNA PLATAFORMA DE CLASIFICACIÓN DE TEXTO BASADA EN TEXTMINING APLICADA SOBRE UNA RED DE BLOGS PARA BETAZETA NETWORKS S.A.

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN
INGENIERO CIVIL INDUSTRIAL

CAMILO ALBERTO LÓPEZ ARAVENA

PROFESOR GUÍA:
SEBASTIÁN RÍOS PÉREZ

PROFESOR CO-GUÍA:
JORGE PÉREZ ROJAS

MIEMBROS DE LA COMISIÓN:
PABLO BARCELÓ BAEZA
CARLOS REVECO DÍAZ

SANTIAGO DE CHILE
MARZO 2012

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN
INGENIERO CIVIL INDUSTRIAL
POR: CAMILO ALBERTO LÓPEZ ARAVENA
PROFESOR GUÍA: SEBASTIÁN RÍOS PÉREZ
FECHA: 05/04/2012

# DISEÑO Y CONSTRUCCIÓN DE UNA PLATAFORMA DE CLASIFICACIÓN DE TEXTO BASADA EN TEXTMINING APLICADA SOBRE UNA RED DE BLOGS PARA BETAZETA NETWORKS S.A.

Betazeta Networks S.A. es una empresa dedicada a la publicación de información mediante una red de blogs de diversas temáticas. A corto plazo, la empresa necesita visualizar cómo se distribuye el contenido actual para tomar decisiones estratégicas respecto al mercado que enmarca los contenidos que publican. En el mediano plazo, la empresa emitirá contenido generado por los usuarios, el cual debe ser revisado para mantener la calidad de cada Blog. Para esto se requiere contar con métodos automáticos de clasificación para dichos mensajes, los cuales serán revisados por periodistas expertos en diferentes áreas.

El trabajo realizado en esta memoria constituye un prototipo que apunta a resolver la problemática de la empresa. Para ello se construye una plataforma de procesamiento de texto, denominada Tanalyzer, que permite manejar grandes volúmenes de información, visualizar, clasificar y hacer predicciones sobre las temáticas de nuevos documentos utilizando text-mining, sub área de la minería de datos especializada en texto, implementando el modelo de tópicos generativo Latent Dirichlet Allocation.

Las pruebas realizadas al software son satisfactorias. Sobre un modelo que maneja 8 temáticas, cada una asociada a uno de los 8 blogs de la empresa que se encuentran bajo estudio, es posible predecir documentos con un 80 % de *precision* y 64 % de *recall*, lo que demuestra la viabilidad de la aplicación.

Actualmente, la solución permite escalar tanto en velocidad como en costos. Con un tiempo de ejecución de 2.5 horas para 300.000 documentos, permite entrenar en ese tiempo un mes de publicaciones a una tasa de 1250 artículos enviados diariamente repartidos en 8 blogs, frente a la tasa actual de publicación de 12.5 artículos diarios por blog. Entrenar 10 veces un modelo de esta magnitud representa para la empresa un costo de $USD 17 utilizando los servicios de Amazon Cloud Computing.

Si bien los resultados obtenidos son positivos y la memoria cumple sus objetivos a cabailidad, existen múltiples mejoras realizables a la plataforma que constituyen el trabajo futuro de esta investigación y que deben ser consideradas por la empresa para llevar a cabo una implementación en producción. Por un lado es posible mejorar aún más los tiempos de ejecución y por otra parte se debe solucionar la disminución de *recall* cuando la cantidad de temáticas y la especificidad de éstas aumenta.

*Agradecimientos*

Más que un proceso de titulación, terminar la memoria significa el cierre de una gran etapa en mi vida.

En primer lugar, nada de esto sería posible sin las enseñanzas de mis padres, Alberto y Julia, pues lo que hoy soy es el reflejo de su esfuerzo en esta tarea. Gracias a Antonieta y María José, por ser no sólo mis hermanas si no que mis dos mejores amigas. Los amo a los cuatro.

Del mismo modo, no puedo dejar de mencionar a mi grupo de amigos de primer año de universidad, los cuales me han acompañado durante todo este tiempo, incluso algunos desde el colegio. Sebastián Águila, Matias Mayol, César Nuñez, Cristian Solís, Sebastián Fuentealba, Nathaly Romero, Felipe Urbina, Gabriela Lazo, Javier Pérez, Eduardo Pérez, Alexis Acuña, María Jesús Rubio y Daniela Arancibia, espero que nos sigamos viendo durante muchos años más. Una mención especial a mi gran amigo Rodrigo Dueñas, quien me acompañó además en el desafío que hoy culmina de sacar dos especialidades de ingeniería.

Debo mencionar también a mis grandes amigos de este último periodo, los cuales se han mostrado siempre incondicionales ante todo. Gracias Juan Muñoz, Matthew de Los Santos, Larry González, Andrés Rebolledo y Felipe Espinoza. Gracias a todos por estar tanto en las buenas como en las malas, además de haber sido fundamentales en mi proceso de aprendizaje y reflexión respecto a la vida presente y futura.

Agradezco a Yury Cancino por idear el proyecto que enmarcó este trabajo, así como también a todo el equipo de tecnología de Betazeta. Una tremenda mención se merecen Cinthya Vergara, Matías Ibañez y nuevamente a mi gran amigo Larry González, los cuales participaron en la implementación de Tanalyzer.

También debo dar las gracias al profesor Richard Weber y sus excelentes cátedras, pues con él hice el curso de Introducción a la Minería de Datos, ramo que conformaría las bases de mi conocimiento en el área, lo que me permitió tomar este desafío.

Finalmente, agradezco a mis dos profesores guías, Sebastián Ríos y Jorge Pérez. Ambos me orientaron en forma clave en los minutos de incertidumbre respecto a cada uno de los dos enfoques que tuve que abordar, transformándose en mis mentores en este proceso.

# Índice General

# Índice de figuras

# Índice de cuadros

# Capítulo 1

# Introducción

Actualmente en el mundo se generan grandes cantidades de información en la Web. Solamente en YouTube se suben 48 horas de video por minuto[1] y en Facebook se comparten 30 billones de piezas de contenido (links, posts de blogs, álbumes de fotos, etc.) cada mes[2]. Por otra parte, dentro de ésta cantidad de información también existe un número importante de datos que no son apropiados. Ejemplos de esto son el bullying virtual, spam, ciber crimen [21] o simplemente contenido fuera de contexto. En 2008 Facebook lanzó un comunicado en su blog oficial[3] donde reconoce la presencia de aplicaciones que envían spam en la red social. De acuerdo a las estadísticas de amenazas globales de Symantec [4], un 81.3 % de los correos enviados en el mundo corresponden a spam. En lo anterior radica la urgencia de moderación en la red. Sin embargo, dado el alto volumen de datos que son producidos día a día, realizar esta actividad en forma manual resulta imposible, por lo que surge la necesidad de desarrollar herramientas que permitan automatizar esta tarea.

Esta memoria planea proveer soluciones a la problemática de la empresa Betazeta Networks S.A., la que en el corto plazo requiere analizar cómo se distribuyen los contenidos que publican y en el mediano plazo abrirá un sistema cuya información será alimentada por usuarios de internet. Betazeta Networks S.A. de ahora en adelante Betazeta, es una empresa dedicada a generar contenidos de nicho específico para publicación en la Web en base a un conjunto de blogs y otros medios online, cuyo modelo de ingresos se basa en la publicidad dirigida a los lectores de la plataforma. Dentro de Latinoamérica y España la empresa ha conseguido un crecimiento sostenido y en la actualidad recibe más de ocho millones de visitas mensualmente a lo largo de su red. Además de lo anterior, existe BetaID, un servicio basado en OpenID[5], ligado a plataformas masivas como Facebook y Twitter, y que permite una identificación única para los usuarios a lo largo de toda la plataforma de blogs.

[1] http://googleblog.blogspot.com/2011/05/youtube-highlights-526.html
[2] http://www.facebook.com/press/info.php?statistics
[3] http://ja-jp.facebook.com/blog.php?post=10199482130
[4] http://www.messagelabs.co.uk/globalthreats/charts/spam_monthly
[5] http://www.openid.net

Actualmente Betazeta cuenta con un número definido de periodistas encargados de generar contenido dirigido por distintas líneas editoriales específicas para cada blog. Cada periodista se encarga de buscar noticias que podrían ser relevantes para el sitio y redacta un artículo, el cual es validado por un editor en términos de calidad y redacción, pero carecen de una visión estratégica asociada al mercado en el nicho que publican. Debido a la cantidad de publicaciones que existen en los distintos blogs, para la empresa es difícil visualizar la segmentación de contenidos que han sido publicados a lo largo de los años. Esto genera dos problemas relevantes:

1. El sitio no es bien indexado por buscadores como Google, ya que presenta una línea de contenidos poco consistente en cada blog.
2. El contenido, en la mayoría de los casos, no está alineado a los intereses del mercado.

A pesar de que la mayoría de los blogs en la red de Betazeta están posicionados como número uno en su segmento, dados los problemas antes mencionados el crecimiento de la empresa, que se traduce en visitantes a sus sitios, está limitado.

Por otro lado, durante el último tiempo se ha detectado que muchos de los lectores que participan en la red de la empresa, a la vez son dueños de blogs de diversas temáticas y también generan contenido. Aprovechando esta situación existen planes de innovar el modelo de negocios, aprovechando la sinergia que ha adquirido la comunidad de usuarios en torno a su plataforma, y abrir la línea periodística al público en el mediano-largo plazo, lo que en teoría debiese permitir aumentar su capacidad para generar artículos de calidad en órdenes de magnitud mayor a lo que se produce actualmente y por ende aumentar sus ingresos en base a publicidad.

De acuerdo a lo expuesto anteriormente, debido a la apertura del equipo redactor es fundamental contar con un mecanismo que permita automatizar, en parte, la labor de clasificación del contenido enviado por los usuarios y también clasificar los contenidos que actualmente se encuentran publicados para poder visualizar la distribución de contenidos y con ello confeccionar una estrategia de mercado más robusta.

## 1.1. Descripción del proyecto

Para Betazeta es relevante poder corroborar si un artículo, enviado a un blog en particular, efectivamente debe ser publicado en dicho blog de acuerdo a su contenido y, por otro lado, es necesario generar un indicador que permita ordenar estas publicaciones seleccionadas de acuerdo a algún criterio. Se espera que este procedimiento en algún minuto deje de ser viable mediante un filtro manual y por tanto sea necesario utilizar un apoyo tecnológico. El proyecto consiste en desarrollar una plataforma que permita en primera instancia clasificar el contenido que se recibe de los usuarios para asociarlo a un determinado blog o eventualmente descartarlo

y, en una etapa posterior, poder evaluar este contenido en forma automática cumpliendo el rol de un primer filtro, complementando el trabajo manual, apoyando a definir: qué contenido es publicado, dónde y en qué orden. De acuerdo a lo anterior, la plataforma debe permitir reducir el volumen de datos recibidos de acuerdo a una clasificación y evaluación a priori, entregando una aproximación de los artículos que posiblemente sean más relevantes para cada blog.

Para hacer funcionar esta plataforma se utilizará minería de datos especializada en texto para analizar los posts publicados por los usuarios inicialmente utilizando los datos históricos que existen en la actualidad. Algunos de los datos disponibles son:

- Información de cuentas de usuario y su interacción en la red como comentarios y tags en blogs y post en foros.
- El contenido de la red propiamente tal que está compuesta por los artículos dentro de los blogs y sus comentarios.
- Número de visitas al contenido y a la publicidad.
- Referencias externas a las publicaciones[6].

En una primera instancia se espera solucionar el problema de clasificación temática, esta primera etapa del proyecto es lo que será abordado en esta memoria. Como segunda etapa del proyecto se investigarán las alternativas posibles para realizar una clasificación de acuerdo a criterios de calidad, problema más complejo ya que los mecanismos dependen de un método particular para cada blog.

En la primera etapa el input a analizar será un texto correspondiente al contenido de un artículo. De aquí se desprende que no se contará con información de visitas ni comentarios o ningún tipo de feedback provisto por el conjunto de usuarios al momento de clasificar un documento ya que el artículo debe ser procesado antes de ser publicado y por tanto, el único elemento disponible para analizar es el propio texto. Para la clasificación esto no representa un desafío mayor, ya que haciendo un análisis sobre las palabras contenidas en dicho texto se cree posible estimar temáticas asociadas con cierta precisión.

## 1.2. Objetivos

A continuación se presenta el objetivo general de este trabajo junto a sus objetivos específicos.

[6]Por ejemplo el número de apariciones en Twitter o "Likes" de Facebook.

### 1.2.1. Objetivo General

El objetivo general del trabajo es el apoyo al procesamiento manual de grandes volúmenes de publicaciones en la red de blogs de Betazeta, mediante el diseño e implementación de una plataforma para visualización de contenidos y la categorización automática de estos datos utilizando text mining.

### 1.2.2. Objetivos Específicos

1. Entender a fondo la problemática y el contexto de la empresa junto con los conocimientos necesarios respecto a text mining, modelos y metodologías necesarias.
2. Seleccionar los datos históricos, los atributos sobre éstos y los modelos que permitan realizar predicciones exitosas de categorización.
3. Establecer métodos y métricas para la evaluación de la solución propuesta.
4. Utilizando el conocimiento adquirido en los objetivos anteriores, diseñar el proceso de categorización automático de posts.
5. Diseñar e implementar un prototipo que permita al usuario ingresar información en forma adecuada para su análisis, procesarla y clasificarla en base a criterios del negocio.
6. Evaluación de la calidad de los resultados utilizando las métricas establecidas.

## 1.3. Resultados Esperados

En función de los objetivos específicos se define la lista de resultados esperados a continuación:

1. La definición del conjunto de datos históricos y los atributos a medir para realizar la investigación, el formato de almacenamiento y el modelo de datos a utilizar.
2. El o los modelos que se utilizarán para probar la clasificación de contenido y el marco teórico asociado a cada uno de estos, explicando la razón de su utilización.
3. Un prototipo que permita realizar pruebas con los métodos previamente definidos de clasificación, realizar una visualización de los datos y categorizar en forma semi-automática[7] nuevos textos.
4. La lista de métricas que permitan evaluar dichos modelos, explicando qué implica cada indicador.

[7] Siempre existirá un porcentaje de error asociado al método que deberá ser corregida en forma manual.

5. Una metodología que defina el procedimiento a seguir para poder ejecutar y validar cada modelo en base a las métricas.

6. La lista de resultados tras la validación del prototipo aplicada sobre los modelos analizando los modelos y configuraciones estudiadas.

7. Plasmar la información que permita a la empresa tomar decisiones estratégicas respecto a los próximos pasos a seguir en la evolución del prototipo, en particular, basándose en los resultados de los experimentos y definir el trabajo futuro necesario para una implementación definitiva.

## 1.4. Alcances

En la actualidad el principal mecanismo para evaluar el contenido en la Web consiste en el input de los usuarios; la efectividad de un post se mide comúnmente en base a su cantidad de visitas, comentarios y sistemas de karma[25]. Lo anterior se podría integrar en un modelo que permita discernir si un nuevo texto podría generar más valor que otro. Como se menciona, para el proyecto de Betazeta es relevante considerar estos factores, los cuales serán integrados en una segunda etapa, pero escapan a los alcances de ésta memoria que representan la primera etapa de la investigación, la que contempla un primer clasificador en base a texto plano.

El idioma a analizar será solamente el español. Esto presenta algunos desafíos ya que los algoritmos de stop words y stemming no están tan desarrollados en este idioma como lo están en inglés. El hecho de tomar en cuenta sólo el español deja fuera de análisis al sitio FayerWayer Brasil que se encuentra escrito en portugués.

El prototipo se limitará a utilizar LDA como mecanismo de generación de modelos predictivos, esto se debe a que el tamaño de la plataforma que se espera construir no es pequeño e involucra complejas etapas de manipulación de texto. Incluir la capacidad de trabajar con más modelos representa un costo que escapa al alcance de este proyecto.

Se destaca que el desarrollo de software se centrará en proveer los prototipos para experimentación y validación los modelos, pero no poseerán las características como para ir directo a la línea de producción y aplicación real sobre alta carga de datos y concurrencia de usuarios. Esto permite expandir el proceso de investigación y deja en manos de Betazeta hacer la aplicación escalable en su plataforma.

# Capítulo 2

# Marco Metodológico

En esta sección se describen las tres metodologías que se utilizan para el desarrollo de ésta investigación. Esto incluye el desarrollo de software, el proceso de minería de texto y la validación de los experimentos realizados.

## 2.1. Metodología para el desarrollo del software

Si bien no se espera lograr un software listo para la utilización en producción, sí es necesario lograr un prototipo suficientemente desarrollado como para confirmar la viabilidad del proyecto y que permita realizar clasificación con un grado de certeza aceptable.

Además, al ser un proyecto de investigación, existe alta incertidumbre sumado a que las necesidades de la empresa varían en el tiempo, generando constantes cambios en los requisitos de la plataforma.

Según lo expuesto, es necesario trabajar utilizando un mecanismo que permita adaptarse a los cambios y poder validar diferentes hipótesis de investigación en un tiempo breve, es por esto que para la confección del software se sigue una *metodología de desarrollo ágil* [9], ver figura 2.1, esta consiste en diseñar pequeños prototipos que cumplan todo el proceso esperado pero con bajo nivel de detalle y funcionalidad y luego iterar sobre éste para verificar el cumplimiento de los requerimientos, anticipando un posible cambio en ellos. El objetivo de cada prototipo es generar valor, minimizando así los riesgos del proyecto, hasta alcanzar un estado deseado en el software.

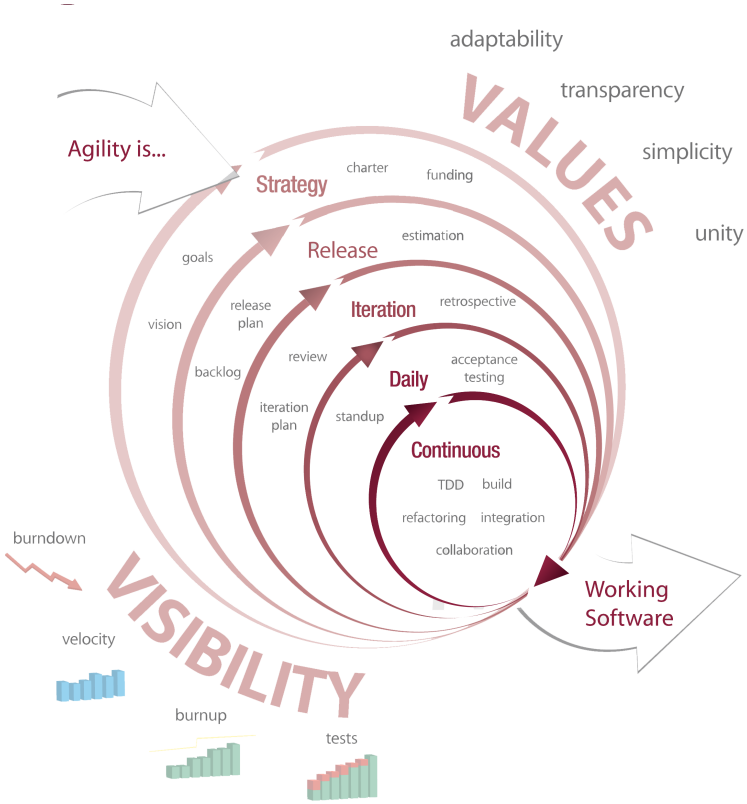


**Figura 2.1:** Fases del desarrollo ágil.

**Fuente:** http://en.wikipedia.org/wiki/Agile_software_development

La metodología de software ágil se prefiere a otras del área dada su amplia flexibilidad además de que permite rápidamente alcanzar un punto de prueba para poder ser validado, muy importante en caso de modificar algún objetivo de la memoria durante su desarrollo, minimizando la cantidad de pérdida de trabajo y poder comenzar a validar lo antes posible los resultados obtenidos con el cliente. Como ya se mencionó, dado que este es un proyecto de investigación, la volatilidad en los requerimientos es sumamente alta, por lo que es conveniente usar una metodología de este tipo para abordar el problema.

## 2.2. Metodología para realizar minería de datos

Durante los últimos años se ha visto un crecimiento y consolidación del área de Data Mining. Por este motivo se han hecho esfuerzos por encontrar estándares en la realización de ésta práctica. Algunos de estos esfuerzos se traducen en el desarrollo de las metodologías

SEMMA[23] y CRISP-DM[8][17]. Ambas metodologías apuntan a definir estándares sobre la implementación práctica del proceso KDD en la industria, definiendo un conjunto de pasos concretos dentro un proceso bien definido.

### 2.2.1. KDD

KDD o Knowledge Discovery in Data [13] corresponde al proceso de realizar minería de datos para obtener conocimiento de acuerdo a la especificación de ciertas medidas y tolerancias, usando un conjunto de datos junto a cualquier proceso, muestreo y transformación requerida sobre éstos.

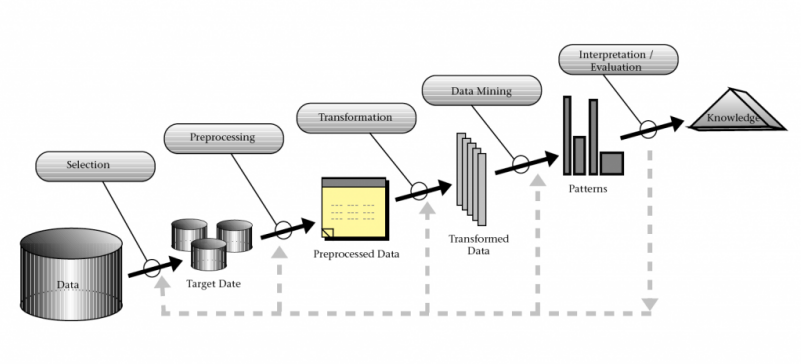


**Figura 2.2:** Proceso de KDD.

**Fuente:** http://bituchile.com/.

En este proceso se consideran normalmente cinco etapas, ver Figura 2.2:

- Selección: Esta etapa consiste en crear un conjunto de datos de trabajo enfocado en un subconjunto de variables o un muestreo de los datos sobre el cual realizar el cual descubrir conocimiento.
- Pre-procesamiento: Consiste en la limpieza de los datos con el objetivo de generar data consistente.
- Transformación: Esta etapa corresponde a reducir la dimensionalidad y hacer los cambios necesarios que permitan su correcta manipulación.
- Minería de datos: Esta etapa consiste en la búsqueda de patrones de interés y una aplicación particular como clasificación o predicción sobre nuevos datos.
- Interpretación y Evaluación: Esta etapa consiste en la interpretación y evaluación de los patrones minados.

Éste proceso es interactivo e iterativo, e involucra numerosos pasos con muchas decisiones hechas por el usuario. El proceso KDD es precedido por el entendimiento del dominio de aplicación del problema, el conocimiento previo relevante y los objetivos del usuario final. Debe ser continuado con la consolidación de conocimiento y la incorporación de éste en el sistema.

A continuación se presentan dos formas de implementar este proceso.

### 2.2.2. SEMMA

El proceso SEMMA (Sample, Explore, Modify, Model, Assess), ver Figura 2.3, se utiliza para administrar proyectos de minería de datos y, como su nombre lo indica, consiste en un ciclo de cinco etapas.

- *Sample* o Muestreo: Consiste en extraer un subconjunto de los datos originales con el fin de tener un set de datos suficientemente representativo pero que permita una manipulación rápida.
- *Explore* o Exploración: Implica comprender el conjunto de datos, buscando tendencias inusuales o anomalías con el fin de profundizar la comprensión de los datos y obtener nuevas ideas.
- *Modify* o Modificación: Es básicamente la extracción de variables, modificación o cualquier cambio sobre los datos que permitan prepararlos para la fase de Modelado.
- *Model* o Modelado: Consiste en modelar los datos permitiendo que un software pueda realizar predicciones sobre un determinado fenómeno, a partir de la data analizada.
- *Assess* o Evaluación: Es la fase en la que se evaluan los resultados y se mide la confiabilidad de éstos.

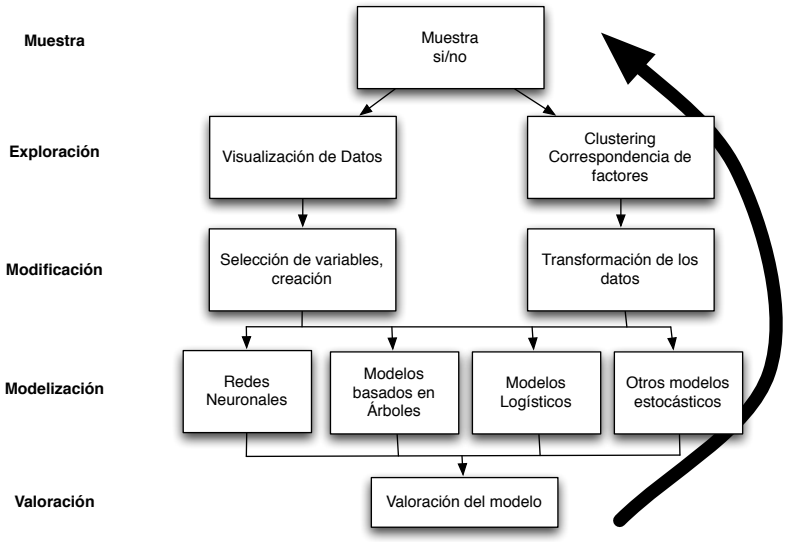


**Figura 2.3:** Metodología SEMMA.

**Fuente:** Elaboración propia a partir de http://www.actuarios.org/.

### 2.2.3. CRISP-DM

CRISP-DM o CRoss Industry Standard Process for Data Mining consiste en un ciclo de seis etapas, ver Figura 2.4:

1. Entendimiento del Negocio: Implica entender los objetivos del negocio, determinar las metas de la minería de datos y definir el plan del proyecto enmarcándolo como un objetivo empresarial.

2. Entendimiento de los Datos: Esta etapa se compone de dos partes: la primera consiste en obtener los datos para el análisis. La segunda implica comprender los datos para identificar problemas de calidad y observar patrones a simple vista que permitan sustentar una hipótesis.

3. Preparación de los datos: Se realizan todos los pasos necesarios que permitan llevar la data original al nivel de calidad apropiado para ser utilizada en los procesos posteriores. Esto puede involucrar selección de atributos, limpieza de datos, construcción de nuevos datos, consolidar datos externos, modificar formatos, etc.

4. Modelado: En esta fase se define el modelo y su input además de diseñar el set de pruebas para su validación posterior.

5. Evaluación: Se aplican los test de evaluación, se define la lista de modelos aprobados y se determinan los pasos que permitan continuar el proceso.

6. Desarrollo: Esta fase corresponde a la etapa final e implica diseñar el plan de desarrolllo, monitoreo y mantención del proceso que utiliza los modelos seleccionados. Además, se realizan los reportes finales y se evalúa el proyecto completo.

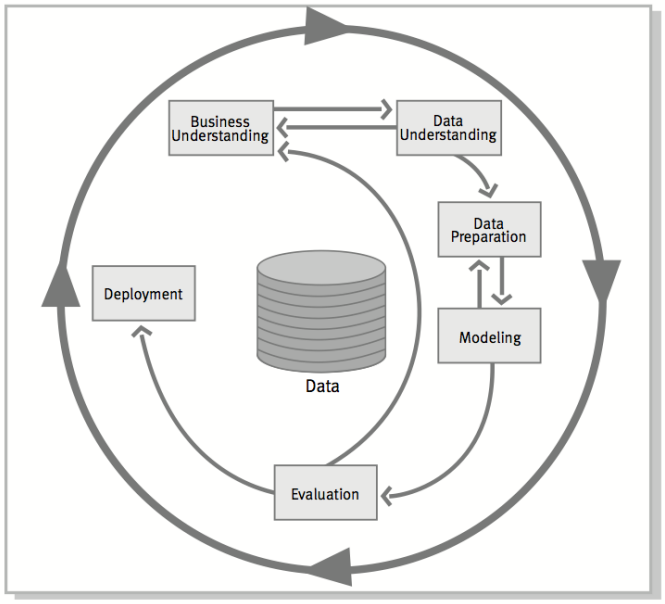


**Figura 2.4:** Pasos a seguir en CRISP-DM.

**Fuente:** CRISP-DM 1.0 Step-by-step data mining guide[8].

Como se observa en las definiciones, ambos métodos permiten una correcta implementación de KDD, ya que las dos metodologías incorporan todas las etapas del proceso. Se prefiere CRISP-DM para este problema en particular, en vista de que se considera más robusto, pues incorpora explícitamente la comprensión de la problemática empresarial, punto muy relevante para Betazeta. Por otro lado, la metodología SEMMA está ligada directamente a la utilización de un software específico [13], el SAS Enterprise Miner Software y, en el caso de este proyecto, se desarrolla un software a medida para análisis de texto por lo tanto es deseable una metodología neutra en este ámbito como CRISP-DM.

## 2.3. Metodología de evaluación

En el marco de este trabajo, se analizarán múltiples documentos a los cuales se les asignará una clasificación utilizando un método no supervisado. Es importante conocer la efectividad de dicho método sobre el conjunto de textos y para ello es necesario definir en qué rangos una respuesta es positiva, ambigua o incorrecta.

Por otra parte, en un problema como el que se aborda en esta memoria, el mecanismo de evaluación, que permite decidir si el resultado de la clasificación de algún método en particular es correcto o no, necesita comparar los resultados del clasificador automático con una fuente confiable de datos, generando lo que se denomina como *Conjunto de Referencia*.

Para abordar los requerimientos antes mencionados se propone una metodología de evaluación diseñada como parte de este trabajo. En lo que resta de ésta sección se explica brevemente el clasificador y el método utilizado para validar los experimentos.

### 2.3.1. El clasificador

Cuando un texto es clasificado el software genera una distribución sobre las clases determinadas en forma no supervisada. Sea C el conjunto de clases determinadas por el algoritmo, con $C = \{c_1, c_2, ..., c_k\}$. Para un texto dado, la distribución de las clases otorgada por el clasificador cumplirá qué:

$$\sum_{z=1}^{k} dc_z = 1 \tag{2.1}$$

$$dc_i \in [0, 1] \ \ \forall i \tag{2.2}$$

Donde $dc_i$ es un número real que representa la distribución de la clase $c_i$ para un texto en particular. A esta distribución se denomina en este trabajo como el *porcentaje de asociación* a una clase.

### 2.3.2. Los rangos de validación

Ahora bien, se busca determinar para qué rangos de $dc_i$ podemos tener certeza de que un texto pertenezca o no a dicha clase, por ejemplo, es útil saber si la categoría *videojuegos*, que posee $dc_{videojuegos} = 0{,}3$ para un texto específico, es con un 90 % de probabilidad la clase principal del texto y la categoría *fiestas*, que posee $dc_{fiestas} = 0{,}01$ en el mismo texto, es con un 95 % de probabilidad una categoría muy alejada a dicho texto. Para esto se define el indicador Quality:

$$Quality = \frac{Vc}{Nw} \tag{2.3}$$

Donde $Vc$ corresponde al valor asignado por el clasificador y $Nw$ representa el número de palabras en el texto analizado. Con lo anterior es posible aproximar cuáles son los rangos de Quality que otorgan certeza de una respuesta y cuáles son aquellos en los que es difícil decidir, esto se analiza en detalle en el Capítulo 5.

### 2.3.3. Diseño de los métodos de clasificación

Para evaluar que un clasificador funciona de manera correcta es necesario generar un conjunto de referencia y para ello existen dos casos. El primero corresponde a asignar una clasificación automática a los documentos basándose en el Blog al que pertenecen. El segundo caso corresponde a clasificar los datos en forma manual, bajo este contexto se procede como sigue:

1. Se presentará al usuario el conjunto de clases generadas durante el proceso de entrenamiento.
2. Se presentará al usuario un texto a evaluar.
3. Se pedirá al usuario seleccionar las clases más representativas o asociadas a dicho texto.
4. Utilizando el algoritmo se determinarán las clases asociadas al texto.
5. Se comparará el resultado del algoritmo respecto a la clasificación del usuario, contando para cada clase el número de coincidencias entre la respuesta del usuario y los rangos entregados por el clasificador.

Ambos casos de profundizan en el Capítulo 5.

# Capítulo 3

# Marco Conceptual

A continuación se mencionan los conceptos necesarios para la correcta lectura de ésta memoria y la comprensión de la solución propuesta.

## 3.1. Blog y Post

Hoy en día las palabras 'blog' y 'post' son ampliamente utilizadas en la Web. De acuerdo a The Economist [12]:

> "El término 'blog' aparece por primera vez en 1997, cuando uno de los practicantes de la época, Jorn Barger, llamó a su sitio un 'Weblog'. En 1999, otro usuario, Peter Merholz, a modo de broma separó la palabra en 'we blog', y de alguna forma el nuevo término –blog- se mantuvo como verbo y sustantivo. Técnicamente se refiere a un sitio Web en el cual su propietario agrega nuevas entradas o 'posts' que tienden a ser cortas y normalmente contienen referencias a otros blogs o sitios Web. Además de texto e hipertexto, un post puede contener fotografías o videos".

## 3.2. Wordpress

En el sitio de Wordpress es posible obtener la siguiente descripción del producto:

> "WordPress es una avanzada plataforma semántica de publicación personal orientada a la estética, los estándares web y la usabilidad. WordPress es libre y, al mismo tiempo, gratuito"

En términos más técnicos, Wordpress es una herramienta open-source [1] diseñada para la administración de *Blogs*, haciendo muy sencillo el proceso de escribir y publicar artículos o *Posts*. Está programado en el lenguaje PHP y permite la integración de plug-ins con el fin de expandir sus funcionalidades básicas.

## 3.3. Minería de datos y texto

En la actualidad existe una gran cantidad de datos que se generan diariamente en diversas áreas como transacciones bancarias, medicina, servicios al cliente, etc. Como respuesta a este hecho aparece el proceso de Knowledge Discovery in Databases o KDD [14], proceso definido anteriormente. La etapa de data mining en éste proceso consiste básicamente en la extracción de patrones en un conjunto de datos [32], mezclando procedimientos de inteligencia artificial y estadística, con la finalidad de poder clasificar o hacer predicciones respecto a los datos al enfrentar altos volúmenes de información[22].

Un área que representa un subconjunto de la minería de datos es la minería de texto o text mining, que corresponde a utilizar el mismo concepto anterior, pero con la especificidad de hacerlo sobre texto en lugar de cifras o números; diferencia clave a la hora de manipular la información, ya que en lugar de procesar sólo atributos bien definidos, se utiliza generalmente el universo de palabras disponibles en un conjunto de textos, aumentando la dimensionalidad del problema en varios órdenes de magnitud[35].

Para tener efectividad, ambos mecanismos, la minería de datos y la minería en texto, requieren utilizar datos cuidadosamente seleccionados, acotados y muchas veces modificados para mejorar la calidad de la información y, en consecuencia, la efectividad de los métodos. Lo anterior corresponde a los procesos de selección, procesamiento y transformación de atributos del proceso de KDD descrito en el capítulo anterior. En el caso de la minería en texto, para el idioma inglés, el proceso se divide típicamente en dos etapas: el proceso de Stemming de palabras y la remoción de Stop Words [20].

## 3.4. Selección de atributos

La selección de atributos en minería en texto es fundamental. Debido a la amplitud de datos a analizar, en este caso un diccionario completo de palabras que aparecen en un conjunto de textos, es imprescindible reducir el tamaño de las muestras ya que muchos de los métodos clásicos aplicables a minería de datos pueden llegar a incurrir en tiempos exponenciales de cálculo. Es por ello que la selección de atributos cumple dos roles: intentar mejorar los niveles de predicción como también los tiempos de ejecución de los algoritmos[19]. A continuación se nombran algunas de las técnicas más relevantes para la selección de atributos en el área de text mining.

[1] http://es.wikipedia.org/wiki/Codigo_abierto/, Wikipedia en sí mismo es un proyecto open-source.

### 3.4.1. Stemming, n-gramas y lematización

Al realizar text mining una técnica habitual es contar la frecuencia de palabras en un texto, pero muchas veces palabras diferentes pueden representar ideas similares y en consecuencia es deseable hacer una normalización. Una manera de realizar éste proceso es hacer Stemming, que busca transformar un grupo de palabras a su raíz morfológica común para poder agruparlas en un solo concepto, minimizando así la dimensionalidad del problema y reduciendo la dispersión en el cálculo de frecuencia de los términos. Uno de los algoritmos más utilizados usando éste proceso es Porter, que se basa en utilizar un set de reglas asociadas a las terminaciones de cada palabra para reducir ésta a una raíz, normalmente común, entre conjugaciones de una misma palabra. Su implementación cuenta con versiones tanto para el español como para el inglés [11].

Para el idioma español, uno de los mecanismos más efectivos es el s-stemming [15]. Originalmente éste método se basa en quitar las "s" finales de todas las palabras para dejar todo en singular. Para el español es posible extenderlo teniendo en cuenta que para sustantivos y adjetivos que terminan en consonante se agrega "es" al plural. En el caso que una palabra en singular termine en "e" es posible generar inconsistencia, por tanto se eliminan también estas letras.

Otro método consiste en la descomposición en n-gramas. Un n-grama representa una parte de n segmentos consecutivos de un string más largo, la idea consiste en obtener todos los n-gramas posibles. En el caso de stemming por n-gramas los segmentos son letras. Por ejemplo, usando bi-gramas, la palabra 'hola' se puede descomponer en "h", "ho", "ol", "la", "a". En general, para bi-gramas, una palabra de largo $K$ se puede descomponer en k+1 bigramas considerando los espacios de inicio y fin que aparecen naturalmente en un texto. A partir de lo anterior es esperable que palabras que pertenecen a la misma raíz posean un alto número de n-gramas en común[15][7]. Se destaca además que éste método también puede ser utilizado directamente para la categorización de texto y no solo para la implementación de stemming.

Otro método utilizado es la *lematización flexiva*, que supone que una palabra flexionada a partir de un lema original llega a ese estado a partir de un conjunto de transformaciones que siguen ciertas reglas acorde al lenguaje. Teniendo las reglas es posible inferir las transformaciones que una palabra sufrió para llegar al estado actual y de ésta forma determinar el término raíz. Este método obtiene resultados cuya precisión se sitúa entre un 95 % y 98 % pero su complejidad de implementación es alta dada las excepciones que existen en lenguajes como el español [15].

Un método ligado a la lematización flexiva es la *lematización derivativa* que consiste en determinar derivados de un lema que no necesariamente corresponden a flexiones de una palabra original. Por ejemplo, para la palabra "chica" existe la derivación "chiquilla". La realización de este método sigue un procedimiento similar al de la lematización flexiva. Se utiliza un conjunto de reglas que aplican sufijos derivativos para determinar la palabra original. Como muchas flexiones podrían considerarse derivaciones, es importante realizar como paso anterior un análisis de lematización flexiva antes de la derivativa [15].

### 3.4.2. Stop Words

De acuerdo a *The automatic identification of stop words*[36],

> "Una stop word corresponde a una palabra que tiene la misma probabilidad de aparecer tanto en aquellos documentos relevantes para una consulta como para aquellos que no."

La remoción de estos términos es un método utilizado ampliamente en recuperación de la información. Ejemplos de lo anterior son palabras como "el", "así", "y", "ellos" o también conectores como "por lo tanto", "es decir", "sin embargo", que aparecen frecuentemente en textos en español. Es posible obtener diccionarios en diversos idiomas, incluso español, en la Web1[2] o generar un conjunto de stop words en base a los textos a analizar observando sus frecuencias de aparición [28].

## 3.5. Representación de texto

Uno de los modelos clásicos para la representación de textos y su análisis corresponde al modelo Vector Space Modeling (VSP) o también conocido como "bag of words" [29] o bolsa de palabras. Este modelo considera que dado un conjunto $t$ de palabras, un documento es representado por una selección de dichas palabras sin importar su orden, por lo tanto, bajo esta premisa "hola, cómo estás" y "estás hola cómo" representan exactamente lo mismo. Formalmente, se define un documento $D$ como un vector de dimensión t:

$$D = (d_1, d_2, d_3, ..., d_t)$$

En el que $d_j$, con $d_j > 0$, representa el "peso" o "relevancia" de un término $j$ en D, con 0 representando mínima relevancia. Un mecanismo para calcular relevancia sobre un conjunto de documentos ($D_1$, $D_2$,..., $D_K$) es TF-IDF o Text Frequency - Inverse Document Frequency, que consiste en que el peso de un término $j$ respecto a un documento $i$ ($D_i$) está dado por:

$$d_{i,j} = n_{i,j} * log(\frac{K}{n_j})$$

En el que $n_{i,j}$ es la frecuencia local del término en el documento $D_i$ (apariciones de la palabra $j$ en el documento $i$ dividido por el total de términos en el documento i), $K$ es el total de documentos y $n_j$ es el número de documentos en el que aparece la palabra j. De ésta manera, el conjunto de $K$ documentos se modela como una lista de vectores ($D_1$, $D_2$,..., $D_K$) donde:

[2]Proyecto Snowball: http://snowball.tartarus.org/algorithms/spanish/stop.txt

$$D_i = (d_{i,1}, d_{i,2}, d_{i,3}, ..., d_{i,t})$$

Con lo anterior se construye una matriz en que las filas representan la relevancia de una palabra en los diferentes documentos y las columnas representan la composición de palabras que forman un documento en particular [1], esto recibe el nombre de Matriz término-documento.

## 3.6. Clasificacion de texto

En el ámbito de la recuperación de la información, en particular para la minería en texto, existen múltiples formas de abordar el problema planteado de clasificación de documentos. Específicamente el problema a resolver consiste en que dado un documento desconocido, el método debe ser capaz de definir su pertenencia a un grupo con cierto grado de certeza utilizando información de otros documentos. En la presente sección se mencionan algunos de los métodos más relevantes para clasificación supervisada, para luego describir uno de los modelos con mayor impacto en clasificación no supervisada, Latent Dirichlet Allocation.

### 3.6.1. Modelos de clasificación supervisados

Un modelo de clasificación supervisado es aquel que aprende de un input, denominado "conjunto de entrenamiento", donde cada entrada posee una clasificación previa. El modelo toma esta información y busca una correlación entre los datos de entrada y la categoría asignada. Utilizando esta información de correlación es posible tomar una nueva entrada cuya clasificación es desconocida e intentar asignar una en base al modelo calculado previamente.

Eric Jiang, en [20] utiliza algoritmos de aprendizaje en máquinas para la clasificación de SPAM. En dicho estudio se trabaja con cinco de las técnicas de aprendizaje supervisado más comunes que son Naïve Bayes, Support Vector Machine (SVM), logitBoost, Augmented Latent Semantic Indexing (LSI) Space Model y redes RBF (Radial Basis Function), donde se concluye que las más efectivas son LSI y RBF. Se destaca en este caso que la clasificación de SPAM es binaria, mientras que el problema de la empresa posee múltiples categorías, lo que lo hace aún más complejo. De acuerdo a [35], otro método más simple de clasificación consiste en medir la separación entre dos textos; uno ya clasificado y otro sin clasificar, usando el algoritmo del vecino más cercano o k-nearest-neighbor (KNN) para construir grupos o clusters utilizando una función que defina distancia entre dos textos, utilizando las palabras que aparecen en éstos. Otro mecanismo estudiado en la misma fuente es la utilización de reglas de decisión (o árbol de decisión) donde en base a una serie de reglas o condiciones el documento puede ser clasificado. Otros estudios [16] utilizan también SVM como método de clasificación complementándolo con factorización de matriz no negativa (NMF) como mecanismo para reducir atributos, con el fin de disminuir el tiempo de ejecución de los algoritmos de clasificación.

El problema de los modelos de clasificación supervisados es que requieren un set de datos previamente clasificados. Al analizar texto muchas veces es necesario hacer esta clasificación

en forma manual lo cual presenta tres dificultades:

1. Si el input necesario requiere muchos documentos, la tarea de categorización manual es costosa.
2. La tarea de clasificación muchas veces va a requerir expertos en el área.
3. La clasificación para los datos del conjunto de entrenamiento puede ser ambigua o subjetiva.

### 3.6.2. Topic Models, introducción a LDA

Además de los métodos clásicos antes mencionados aparece la rama de topic models que busca encontrar temáticas implícitas en un conjunto de textos. Esta área nace a partir del análisis semántico latente (LSA) [10] como respuesta a la simplicidad del modelo bag of words descrito anteriormente [29]. Éste método busca reducir la matriz de término-documento, difícil de procesar debido a su tamaño, a un subespacio k-dimensional que capture la mayor parte de la varianza en un set de textos y por ende poder agruparlos. Lamentablemente, con algunas eurísticas, este método es costoso computacionalmente y el espacio resultante es difícil de interpretar ya que corresponde a una combinación lineal de un conjunto de palabras del vocabulario original.

Dentro de investigaciones posteriores se propone como alternativa analizar el conjunto de palabras desde un punto de vista probabilístico, surgiendo así probabilistic LSA[18]; modelo generativo que asocia a cada documento d una variable de clase $z_i$ (que representa algún tipo de categoría asociada al documento). Cada categoría se representa como una distribución de palabras con cierto valor para dicha clase, $p(w|z)$. Luego el modelo se parametriza de acuerdo a la distribución conjunta de un documento $d$ y una palabra $w_i$ como sigue:

$$p(d|w_i) = p(d)p(w_i|d) \tag{3.1}$$

$$p(w_i|d) = \sum_{z=1}^{k} p(w_i|z)p(z|d) \tag{3.2}$$

Debido a que el modelo no es robusto, en la práctica hay una tendencia al sobreajuste de modelos basados en pLSA, ya que no es realmente un modelo generativo completo [1].

Con el objeto de mejorar las debilidades de pLSA se gatilla el estudio de Topic Models, área que busca descubrir variables ocultas utilizando análisis bayesiano jerárquico. Dentro de

los modelos planteados se propone Latent Dirichlet Allocation (LDA) [6]. En este modelo se asume que un texto normalmente habla de distintas temáticas, cada una en menor o mayor proporción. Por ejemplo, en un texto de medicina que hable sobre nuevas tecnologías aplicadas al campo de la genética aparecerían palabras como 'computador', 'datos' o 'software' asociadas a la parte del texto relacionado al "análisis de información", por otro lado, palabras como 'gen','adn' o 'laboratorio' estarían asociados al área de "ingeniería genética". Intuitivamente se observa que habrán distintos subsets de palabras dentro del texto que pertenecerán a una u otra temática y esto es posible generalizarlo a cualquier conjunto de documentos.

Bajo este enfoque los datos se observan a partir de un *modelo generativo probabilístico* que contiene "variables visibles" y "variables ocultas" o "variables latentes". En el caso de textos, las variables visibles corresponden al conjunto de textos y sus palabras, las variables ocultas son: La relación entre temáticas-documento y palabra-temáticas, y que se encuentran implícitas. El hablar de un proceso generativo implica que el conjunto de documentos es posible generarlo mediante un procedimiento con ciertos supuestos donde las variables ocultas son conocidas. LDA asume que los documentos a analizar son definidos en base a este modelo generativo y busca aplicar "ingeniería reversa" para poder inferir cuáles fueron las variables ocultas utilizadas.

Formalmente definimos el *vocabulario* como el conjunto completo de palabras únicas que aparecen en un conjunto de documentos $D$. Una *temática* se define como una distribución de probabilidad sobre las palabras del *vocabulario*. Por ejemplo, el tema "Análisis de Información" tendrá una distribución que asigne una alta probabilidad a las palabras como 'datos' y 'software' y baja probabilidad a 'gato' o 'casa'. Para el proceso generativo se asume que las distribuciones de palabras para cada temática están previamente dadas.

El modelo generativo para cada documento puede ser descrito en dos fases [3]:

1. Seleccionar una distribución de probabilidad aleatoria sobre las *temáticas* definidas.
2. Para cada palabra que se va a colocar en el documento:
    - Seleccionar una temática en forma aleatoria a partir de la distribución de probabilidad definida en el primer paso.
    - Seleccionar una palabra en forma aleatoria a partir de la distribución de la temática seleccionada.

El paso uno define las temáticas que compondrán el texto. El paso dos se encarga de "escribir" el texto, seleccionando palabras asociadas a las temáticas elegidas.

Para determinar las estructuras ocultas se utiliza inferencia posterior. Básicamente se intenta responder la pregunta *¿Qué temáticas me permiten describir una determinada colección de textos?*.

### 3.6.3. Latent Dirichlet Allocation

Como ya se mencionó, LDA pertenece a la familia de modelos generativos probabilísticos que incluyen variables ocultas. En este tipo de modelos el proceso generativo define una distribución conjunta de probabilidad sobre las variables ocultas y observadas. Utilizando esta distribución se intenta calcular la probabilidad condicional de las variables ocultas dadas las variables observadas, la que se conoce también como la distribución a posteriori.

Dado lo anterior es posible definir LDA formalmente como sigue [6]:

- Las temáticas estarán dadas por $\beta_{1:K}$, donde cada $\beta_k$ representa una distribución sobre el vocabulario.
- La distribución de temáticas para un documento $d$ está dada por $\theta_d$ en el que $\theta_{d,k}$ es la proporción para la temática $k$ en el documento $d$.
- La asignación de temáticas para el documento $d$ se define como $z_d$ en el que $z_{d,n}$ es la asignación de temática para la palabra $n$ dentro del documento $d$.
- Las palabras observadas del texto $d$ se representan por $w_d$ en el que $w_{d,n}$ es la n-ésima palabra del documento $d$.

Con lo anterior el proceso generativo de LDA se define como la siguiente distribución conjunta [6]:

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{i=1}^{K} p(\beta_i) \prod_{d=1}^{D} p(\theta_d)(\prod_{n=1}^{N} p(z_{d,n}|\theta_d)p(w_{d,n}|\beta_{1:K}, z_{d,n})) \tag{3.3}$$

Uno de los aspectos relevantes que definen este modelo es la dependencia que existe entre sus distintos parámetros. Cada palabra observada en el documento, $w_{d,n}$ depende de la asignación de tópico $z_{d,n}$ y de la distribución temática $\beta_{i:K}$. De la misma forma, las asignaciones de tópicos $z_{d,n}$ dependen de la distribución de temáticas en el documento d, $\theta_d$. Gráficamente esto se puede ver en la figura 3.1 donde los nodos representan variables aleatorias. La orientación de las aristas entre nodos indican una relación de dependencia. Las variables visibles están en gris. Cada rectángulo en el diagrama indica re-utilización de parámetros, por ejemplo $N$ define la distribución de palabras mientras que $D$ describe el set de documentos que implícitamente contiene a $N$. $K$ en ese sentido es independiente. $\eta$ y $\alpha$ son constantes que que alimentan las distribuciones $\beta_k$ y $\theta_d$ respectivamente.

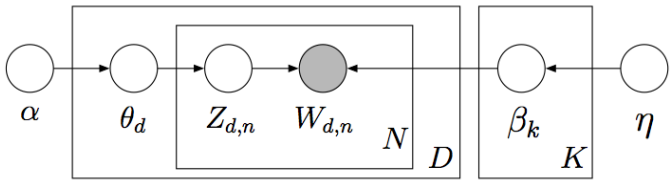


**Figura 3.1:** Modelo LDA.

**Fuente:**David Blei [6].

De acuerdo a la notación explicada anteriormente, el cálculo de la posteriori, de acuerdo a los documentos observados, está dada por:

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}|w_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})} \tag{3.4}$$

El numerador del término anterior corresponde a la distribución conjunta. Si se fijan los parámetros ocultos, entonces éste término puede ser calculado utilizando la ecuación 3.3. Por otra parte, el denominador, que corresponde a la *probabilidad marginal* de las observaciones, representa la probabilidad de tener un determinado set de documentos bajo cualquier modelo. Este término en la práctica es incalculable, ya que su costo computacional es muy alto. Para determinar su valor sería necesario calcular la suma de las distribuciones conjuntas para todas las instancias posibles de las variables ocultas. Como esto no es posible, se utilizan métodos que aproximan el valor de este término, denominado *la evidencia*.

Existen dos formas generales de calcular la probabilidad a posteriori en el área de topic models, algoritmos basados en muestreo y algoritmos variacionales.

Los algoritmos de muestreo o *sampling* intentan obtener una aproximación con una distribución empírica. El algoritmo más utilizado para esta tarea corresponde a *Gibbs Sampling*, una forma de *Markov chain Monte Carlo* o *MCMC*.

MCMC corresponde a un conjunto de técnicas diseñadas para generar una muestra de valores para distribuciones probabilísticas con alta dimensionalidad difíciles de calcular. Gibbs Sampling aproxima una distribución muestreando subconjuntos de variables con baja dimensionalidad, donde cada subconjunto está condicionado por los valores de los otros. El proceso se realiza iterando secuencialmente hasta que la aproximación converge a un valor cercano a la distribución original. La calidad del método está asociado a la cantidad de iteraciones que se utilizan [30].

Los algoritmos variacionales constituyen una alternativa determinística ante los algoritmos de muestreo. En lugar de realizar una aproximación para determinar la probabilidad a poste-

riori proponen un conjunto de distribuciones sobre la estructura oculta y luego encuentran la más apta utilizando la divergencia de Kullback-Leibler como medida de distancia entre dos distribuciones de probabilidad [3]. En otras palabras transforman el problema de inferencia a un problema de optimización.

Es importante destacar que LDA es un modelo no supervisado, por lo tanto sólo necesita el texto para poder hacer una inferencia acertada de las variables ocultas en los documentos a analizar. Su mayor ventaja es que las variables ocultas representan la estructura temática de dichos documentos, lo que permite realizar una clasificación en forma automática, tomando en cuenta que realizar esta tarea en forma manual es muy costoso.

### 3.6.4. Modificaciones a LDA

LDA es una herramienta potente a la hora de determinar variables ocultas en grandes volúmenes de texto. Una de sus principales ventajas es que al ser formulado como un modelo probabilístico puede ser utilizado como un módulo en otros modelos más complejos con propósitos específicos.

Nuevos usos para LDA han aparecido a partir de relajar ciertos supuestos que propone el modelo. Uno de los supuestos relevantes que el modelo asume es que el orden de los documentos no es relevante, esto no es necesariamente cierto. Sobre todo en internet, ciertas temáticas rápidamente evolucionan en el tiempo, por ejemplo, nuevas tendencias en el uso de aplicaciones sociales podrían variar mucho en cuestión de meses y por ende, temáticas asociadas a estas tendencias podrían variar radicalmente su vocabulario. Como respuesta a esta situación se desarrolla *dynamic topic model*[4], modelo que considera la variación en el tiempo y por lo tanto toma en cuenta el orden de los documentos. Como resultado, en lugar de entregar sólo una distribución de palabras que representan un tema, el modelo entrega un conjunto de distribuciones para representar la evolución de una temática.

Algo similar ocurre con el orden de las palabras dentro de un documento. LDA trabaja con el modelo "bag of words", que si bien considera frecuencias, ignora la posición de las palabras. En muchos casos este supuesto es muy fuerte y no aplica a un contexto real. En *Topic modeling: beyond bag-of-words*[33] se toma en cuenta este problema y se incorpora al modelo asumir que las temáticas generan las palabras condicionadas a la palabra anterior.

Otro supuesto en LDA es creer el número de temáticas conocido y fijo. *Bayesian nonparametric topic model*[31] propone que el número de temáticas se determina a partir del conjunto de textos durante la inferencia posterior y supone que nuevos documentos pueden tener temáticas antes no vistas. En este caso las temáticas se extienden a un sistema de orden jerárquico que conforma un árbol de temas, que va de lo mas general a lo mas concreto, cuya estructura particular se infiere de los datos.

### 3.6.5. Predicción de categorías

De acuerdo a David Blei[6], un buen método para poder hacer una predicción sobre un nuevo texto utilizando el modelo obtenido mediante LDA es Naïve Bayes. En este marco existen dos grandes aristas al clasificar texto utilizando este método: Naïve Bayes Bernoulli Multivariado y Naïve Bayes Multinomial. De acuerdo a *A Comparison of Event Models for Naive Bayes Text Classification*[24] éste último es mucho más efectivo, por lo tanto será aplicado al clasificador de la aplicación.

#### Naïve Bayes

En el marco de Naïve Bayes, en adelante NB, el proceso de clasificación consiste en determinar cuál es la clase de pertenencia más probable de un documento utilizando sus palabras. NB asume que los documentos son generados a partir de un modelo parametrizado por $\theta$. El modelo está compuesto por una mezcla de componentes $c_j \in C = \{c_1, ..., c_{|C|}\}$, cada componente es representada por un subconjunto disjunto de $\theta$. A partir de las definiciones anteriores, un documento $d_i$ es creado seleccionando una componente de acuerdo a $P(c_j|\theta)$ y luego la mezcla de componentes generan el documento de acuerdo a sus propios parámetros, con la distribución $P(d_i|c_j;\theta)$. Es posible definir la función de verosimilitud del documento utilizando la suma de la probabilidad total sobre todas las componentes:

$$p(d_i|\theta) = \sum_{j=1}^{|C|} p(c_j|\theta)p(d_i|c_j;\theta) \tag{3.5}$$

En términos simples, un documento pertenece a un subconjunto de clases que se encuentran en $C$. Llevando las definiciones anteriores a términos de LDA, estas clases se pueden considerar como las temáticas de un texto. Luego la probabilidad de que un documento se encuentre en $\theta$ está dado por la sumatoria de la probabilidad de que una temática esté en un determinado modelo multiplicado por la probabilidad de que el documento $d_i$ pertenezca a dicha temática, $c_j$.

#### Naïve Bayes Multinomial

En el modelo NB Multinomial se considera la frecuencia de las palabras en el modelo, a diferencia del modelo NB Bernoulli Multivariado, que considera sólo si la palabra aparece o no en un texto, pero no la cantidad de veces. En este marco, un documento es representado de acuerdo al modelo "bag of words" mencionado anteriormente.

Sea $N_{i,t}$ el número de veces que la palabra $w_t$ aparece en el documento $d_i$ y V el vocabulario completo de palabras que aparecen en los documentos, entonces, de acuerdo a [24], la probabilidad de que un documento $d_i$ pertenezca a una clase $c_j$ en el modelo $\theta$ esta dada por:

$$p(d_i|c_j;\theta) = p(|d_i|)|d_i|!\prod_{t=1}^{|V|}\frac{p(w_t|c_j;\theta)^{N_{i,t}}}{N_{i,t}!} \tag{3.6}$$

**Utilización de NB Multinomial con LDA**

Es directo observar que LDA nos permite obtener $p(w_t|c_j;\theta)$, ya que representa la probabilidad de que una palabra pertenezca a la clase $c_j$, si se considera que las clases son las temáticas del modelo LDA. $N_{i,t}$ es posible obtenerlo en forma directa contando las frecuencias de cada palabra.

Ahora bien, cuando se clasifica un documento, para cada clase $c_j \in C$ se calcula la probabilidad anterior y luego los valores se comparan entre sí para determinar cuál es la clase con mayor peso en el documento. De la fórmula se desprende que el término $p(|d_i|)|d_i|!$ es una constante entre las clases, por lo cual es posible omitirlo, simplificando el cálculo a:

$$p(d_i|c_j;\theta) = \prod_{t=1}^{|V|}\frac{p(w_t|c_j;\theta)^{N_{i,t}}}{N_{i,t}!} \tag{3.7}$$

En la ecuación 3.7, muchas probabilidades condicionales se multiplican, una por cada palabra, lo que puede resultar en números muy pequeños y por ende se puede tener pérdida de precisión. Para compensar este problema es mejor realizar los cálculos computancionales sumando los logaritmos de las probabilidades en lugar de su multiplicación. En este caso, la clase o topic con el valor más alto en la suma de logaritmos será equivalente a la clase cuya multiplicación de probabilidades sea mayor, pues la función logaritmo es monótona. Esto se traduce en la siguiente función:

$$p(d_i|c_j;\theta) = \sum_{t=1}^{|V|} log(\frac{p(w_t|c_j;\theta)^{N_{i,t}}}{N_{i,t}!}) \tag{3.8}$$

Por otro lado, para cualquier palabra que no se encuentre en el texto, $N_{i,t}$ será 0, esto hace que su impacto en el cálculo de la probabilidad de pertenencia a la clase $c_j$ sea nulo,

por lo tanto $|V|$ se debe tomar como el set de palabras que pertenecen exclusivamente al texto en lugar del diccionario completo, esto es relevante, ya que a la hora de implementar un algoritmo elimina operaciones innecesarias.

De la misma forma hay que tener en cuenta que cualquier palabra cuya probabilidad de pertenencia a la clase $c_j$ sea 0, provoca que toda la pertenencia a la clase $c_j$ del documento también se anule. LDA toma en cuenta este tipo de problemas, por lo que asigna siempre un peso ínfimo, mayor a 0, a las palabras que no poseen relación a una clase.

## 3.7. Medidas de efectividad

Para evaluar modelos clasificadores comúnmente en la literatura [35] [20] [28] se trabaja con tres medidas de efectividad: *Precision*, *Recall* y *F-Measure*, que se definen como sigue.

$$precision = \frac{\#predicciones\ positivas\ correctas}{\#predicciones\ positivas} \tag{3.9}$$

$$recall = \frac{\#predicciones\ positivas\ correctas}{\#documentos\ de\ clase\ positiva} \tag{3.10}$$

$$F - measure = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \tag{3.11}$$

Otra forma de definir *Precision* y *Recall* consiste en utilizar los términos *falsos positivos y negativos* y *verdaderos positivos y negativos*. Falso negativo o *fn* corresponde al caso en el que un resultado correcto es interpretado como incorrecto, y falso positivo *(fp)* corresponde al caso en el que un resultado incorrecto es interpretado como correcto. Verdaderos positivos *(vp)* y negativos *(vn)* son aquellos casos en el que los datos se interpretan correctamente, es decir, se interpreta un valor negativo cuando la clase es negativa y positiva cuando la clase es positiva. De acuerdo a estos cuatro términos se puede definir *Precision* y *Recall* como sigue:

$$precision = \frac{vp}{vp + fp} \tag{3.12}$$

$$recall = \frac{vp}{vp + fn} \tag{3.13}$$

*Precision* indica el porcentaje de elementos clasificados correctamente respecto al total de elementos clasificados, por otra parte, *recall* indica el porcentaje de elementos correctamente clasificados respecto del total de elementos de clase positiva.

Cuando *Recall* y *Precision* tienen un valor de 1, implica que todos los elementos clasificados como positivos son realmente positivos y no quedó ningún elemento positivo fuera de ésta clasificación. *F-Measure* agrupa ambas medidas en un sólo indicador, otorgando una buena forma de visualizar la calidad de un método particular.

De acuerdo a estas fórmulas, *Precision* y *Recall* disminuyen cuando aumentan los falsos positivos y falsos negativos respectivamente. Son estas tres medidas las que se utilizarán para evaluar la efectividad de los modelos de clasificación.

## 3.8. Desarrollo de software

A continuación se describe el lenguaje Python y algunos conceptos básicos asociados al desarrollo de software.

### 3.8.1. El lenguaje Python

De acuerdo al sitio web de Python[3], éste es un lenguaje dinámico, open-source, robusto y rápido. Actualmente es utilizado en una amplia gama de aplicaciones. Algunas características de Python son:

1. Python respecto a lenguajes como PHP, Java o C posee una sintaxis simple y elegante, generando código legible. En esto es muy similar a Ruby.
2. Es un lenguaje orientado a objetos, posee manejo de funciones como objetos de primera clase (lambdas) y aspectos avanzados como decoradores de funciones.
3. Posee una amplia gama de librerías que permiten extender sus funcionalidades a prácticamente cualquier área requerida.
4. Posee una *shell* de comandos donde es posible correr fragmentos de código, muy útil para hacer pruebas de una funcionalidad particular.
5. Es simple extender python implementando mósdulos sobre lenguajes compilados como C o C++.
6. Es multi-plataforma.

[3] http://python.org/about/

### 3.8.2. Modelo Vista Controlador

Modelo Vista Controlador o MVC [26] es un patrón de diseño de software que abstrae las componentes del desarrollo a tres capas: El modelo, la vista y el controlador.

La vista corresponde al componente del software que interactúa con el usuario y se encarga de desplegar los contenidos y mostrar una interfaz.

El modelo se encarga de obtener los datos requeridos para desplegar en una vista. Es aquí donde se definen mecanismos para interactuar con una bases de datos, archivos de texto, servicios web, etc.

El controlador es el encargado de realizar la lógica de la aplicación, esto quiere decir que se encarga de recibir una petición de un cliente, pedir los datos requeridos invocando los modelos correspondientes y luego enviar estos datos a una vista particular.

### 3.8.3. XML, JSON y Web Service

De acuerdo a la W3C[4], XML o Extensive Markup Lenguage es un formato de texto simple y flexible diseñado para cumplir con los desafios de la publicación electrónica de datos a gran escala.

De acuerdo a JSON.org[5], JSON o Javascript Object Notation es un formato muy ligero basado en la notación de objetos del lenguaje Javascript. Esto se traduce en la serialización de un objeto o cualquier tipo de información a un formato estándar de texto plano.

De acuerdo a la W3C[6] un Web Service corresponde a un software diseñado para soportar interoperabilidad entre máquinas sobre una red utilizando la interfaz WSDL (Web Services Description Lenguage). Otros sistemas se comunican a un web service utilizando típicamente SOAP-messages, usando el protocolo HTTP y una serialización del formato XML.

En la actualidad se han desarrollado alternativas a la serialización de documentos XML y en su lugar se utiliza la serialización de texto en formato JSON, ver más detalles en la documentación de Web Services de Yahoo [7].

### 3.8.4. Cloud Computing y Amazon EC2

Cloud Computing o Computación en la Nube [8] corresponde a un paradigma relativamente reciente que hace referencia a la generación de servicios mediante Internet. Cloud Computing

[4] http://www.w3.org/XML/
[5] http://www.json.org
[6] http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/
[7] http://developer.yahoo.com/javascript/json.html
[8] http://www.itnews.ec/marco/000035.aspx

busca ofrecer todo tipo de prestaciones que un sistema computacional local puede entregar pero empaquetado en servicios on-line. Este paradigma busca encapsular procesos que podrían ser complejos a nivel local y ofrecerlos al usuario en un formato más simple, flexible y escalable. Ejemplos de esto son sistemas de almacenamiento de archivos en la nube o servidores virtuales gestionados en forma remota.

Amazon EC2 [9] es un servicio de Cloud Computing que permite levantar servidores para correr aplicaciones web en forma dinámica, especificando la cantidad de recursos disponibles (memoria ram, procesamiento, almacenamiento, etc), lo que hace posible que el servidor pueda escalar a medida, en función de la carga en el sistema. En la infraestructura EC2 es posible arrancar diferentes instancias de un servidor en minutos y en forma remota aliviando la necesidad de gestionar máquinas físicas. EC2 permite que la arquitectura de hardware de un proyecto tenga costos variables en lugar de altos costos fijos.

Amazon provee tres configuraciones, de acuerdo a las necesidades de trabajo: Small Instance, Large Instance y Extra Large Instance, cuyas configuraciones se listan en la tabla 3.1. De acuerdo al listado de precios disponible en el sitio de amazon [10] se confeccionó la tabla 3.2.

### 3.8.5. Framework y Django

Un framework[27] define un conjunto de buenas prácticas, metodologías y herramientas para encapsular un proyecto, que permiten simplificar algunos aspectos del trabajo y estandarizar otras, haciendo un desarrollo de software más rápido y sostenible en el tiempo.

Un framework se distingue de un conjunto de librerías, porque sigue cuatro principios fundamentales:

1. La orientación del desarrollo es regida por el framework y no por los desarrolladores, estos se deben adaptar a la filosofía de trabajo de un framework que usualmente corresponden a un conjunto de buenas prácticas.
2. Poseen un conjunto de configuraciones y acciones por defecto, que deben ser de utilidad en el desarrollo.
3. El usuario debe poder extender el framework.
4. El código base del framework no se debe poder modificar, lo cual es distinto de extenderlo.

Django[11] es un framework para el lenguaje Pyhton orientado al desarrollo de aplicaciones

[9] Amazon Elasltic Compute Cloud, http://aws.amazon.com/es/ec2//182-1723165-8315650/
[10] http://aws.amazon.com/es/ec2/pricing/
[11] https://www.djangoproject.com/

| **Small** |
|---|
| 1,7 GB de memoria<br>1 unidad informática EC2<br>(1 núcleo virtual con 1 unidad informática EC2)<br>160 GB de almacenamiento de instancias<br>Plataforma de 32 bits<br>Rendimiento de E/S: moderado<br>Nombre de API: m1.small |
| **Large** |
| 7,5 GB de memoria<br>4 unidades informáticas EC2<br>(2 núcleos virtuales con 2 unidades informáticas EC2 cada uno)<br>850 GB de almacenamiento de instancias<br>Plataforma de 64 bits<br>Rendimiento de E/S: alto<br>Nombre de API: m1.large |
| **Extra Large** |
| 15 GB de memoria<br>8 unidades informáticas EC2<br>(4 núcleos virtuales con 2 unidades informáticas EC2 cada uno)<br>1.690 GB de almacenamiento de instancias<br>Plataforma de 64 bits<br>Rendimiento de E/S: alto<br>Nombre de API: m1.xlarge |

**Cuadro 3.1:** Configuraciones de Amazon EC2.

**Fuente:** www.amazon.com.

| **Configuración** | **Linux/UNIX** | **Windows** |
|---|---|---|
| Small | $USD 0.085/hora | $USD 0.12/hora |
| Large | $USD 0.34/hora | $USD 0.48/hora |
| Extra Large | $USD 0.68/hora | $USD 0.96/hora |

**Cuadro 3.2:** Precios de Amazon EC2.

**Fuente:** www.amazon.com.

Web que aplica el principio DRY[12]. Utiliza una variación conceptual del modelo MVC. En el paradigma de Django el *controlador* recibe el nombre de *vista*, el *modelo* mantiene su nombre y una vista recibe el nombre de *template*. En lo que sigue de ésta sección se hará referencia a estos términos acorde la definición de Django, por lo tanto al hablar de "vista" se describe lo que en el modelo MVC común se conoce como "controlador", ver Figura 3.2.

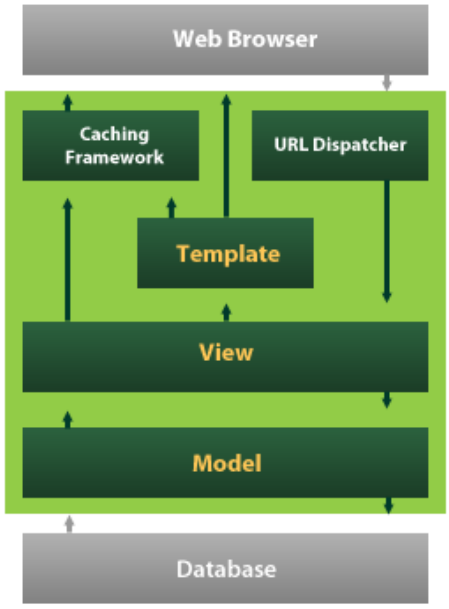


**Figura 3.2:** Arquitectura básica del framework Django.

**Fuente:** https://docs.djangoproject.com/en/1.3/.

Cada software desarrollado usando Django conforma un *project* o proyecto. Cada proyecto a su vez se compone por un conjunto de *applications* o aplicaciones. Una aplicación representa una componente del software. Dentro de un proyecto web estándar una de las aplicaciones podría ser el *frontend*, es decir, la parte que ven los usuarios y otra aplicación el *backend*, la parte que ve el administrador del sitio web.

Cada aplicación dentro de un proyecto posee un conjunto de modelos, vistas y otras configuraciones propias y están pensadas para funcionar de manera independiente. En caso de ser necesario existe la posibilidad de hacer referencia a modelos o vistas entre aplicaciones, ya que es común que una entidad creada en uno de los modelos se requiera en otra aplicación, pues son parte de un mismo proyecto.

[12]http://c2.com/cgi/wiki?DontRepeatYourself

Django posee un sistema de ORM[13], un ORM u Object Relational Mapping consiste en un conjunto de clases que permiten representar una entidad relacional - en la práctica una tabla de una base de datos - como un objeto que hereda sus propiedades y se puede utilizar en el código, creando, actualizando o eliminando elementos en la base de datos.

Cuando se crean nuevos modelos dentro de una aplicación Django extiende el ORM para interactuar con las entidades particulares del proyecto. La conexión estándar de Django a la base de datos es mediante éste ORM, sin embargo, es posible optar por hacer consultas directas a la base de datos, lo cual resulta más eficiente al trabajar con grandes volúmenes de datos, ya que el ORM carga relaciones y ejecuta procesos adicionales para facilitar el desarrollo, pero incrementan la carga en memoria.

Django posee un administrador de urls que permite asociar una dirección web dentro del servidor a una vista específica. Esto permite controlar el acceso a los servicios disponibles en un primer nivel.

Otra de las características de Django es que incluye un servidor dentro del framework. Este servidor normalmente se utiliza para efectos de desarrollo, ya que no permite concurrencia. Usualmente se utiliza otro servidor dedicado a levantar una aplicación a producción como Nginx[14] o Apache[15]. Además de lo anterior, es posible acceder a una *shell* especial de Django, donde se puede ejecutar código directamente, muy útil para probar funcionalidades aisladas (ver figura 3.3). También es posible sincronizar la base de datos en función de la definición de modelos del proyecto, es decir, crear tablas y actualizar el modelo de datos sin acceder al motor de base de datos utilizando Django.

[13] http://www.agiledata.org/essays/mappingObjects.html
[14] http://nginx.org/
[15] http://www.apache.org/

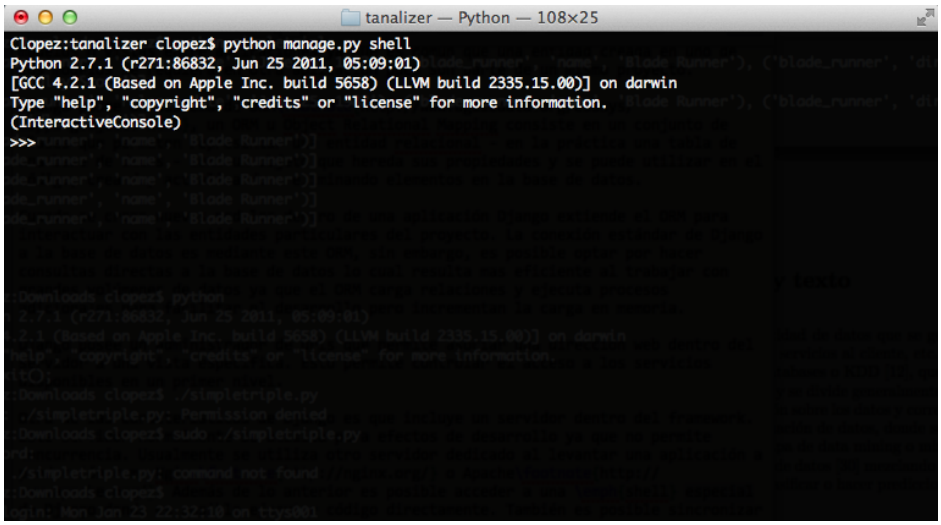


**Figura 3.3:** Shell de Django en uso.

**Fuente:** Elaboración propia.

Finalmente, Django posee un robusto sistema de caché. Cuando un sitio web recibe una petición, en general es mucho más costoso generar el contenido dinámicamente, por ejemplo, leyendo datos desde una base de datos o realizar cálculos matemáticos en comparación a simplemente leer archivos estáticos. Es en este caso cuando el sistema de caché permite reducir la carga, en caso de que existan muchas solicitudes a un contenido específico. A nivel de una vista se pregunta si la url posee un caché, en caso de existir se lee el contenido almacenado en lugar de ser calculado durante la petición.

## 3.9. Blogs a analizar

Si bien no representan conceptos claves y la herramienta que se espera construir tiene un enfoque genérico, es muy importante conocer el contenido de los blogs que serán analizados pues permiten abordar elementos importantes en el aspecto del negocio de la empresa además de comprender las diferentes estructuras de tópicos que se podrían obtener a partir de cada sitio. En la siguiente sección se describirán los blogs analizados en esta memoria, listados en orden alfabético.

### 3.9.1. Belelú

Este es un blog enfocado a público femenino, ver figura 3.4. En http://www.betazeta.com es posible encontrar una descripción para cada uno de los blogs dentro de la red. La descripción para Belelú es la siguiente:


**Figura 3.4:** Blog Belelú.

**Fuente:** www.belelu.com.

> "Una comunidad para mujeres independientes que saben lo que quieren. Mujeres de mundo que se atreven con nuevas ideas, que dan su opinión, que no tienen miedo. Que mantienen un estilo, que buscan tendencias, y que aman la moda. Mujeres, por siempre, muy femeninas".

Además, en cada blog de Betazeta es posible ver en su encabezado las temáticas principales que pertenecen a cada blog. En el caso de Belelú la editorial del blog segmentó sus contenidos en las siguientes categorías: Moda, Sexo y más, Actualidad, Estilo de Vida, Belleza, Entrevistas y Mi Experiencia.

Algunos de los titulares de publicaciones en este blog son:

- "¿Con qué famoso le serías infiel a tu pareja?"
- "Justicia argentina le niega el aborto a niña de 11 años"
- "Mi despedida de soltera ideal"
- "Sexo poco privado"

### 3.9.2. Bólido

La temática de este blog son los autos, ver figura 3.5. La descripción de este blog es la siguiente:

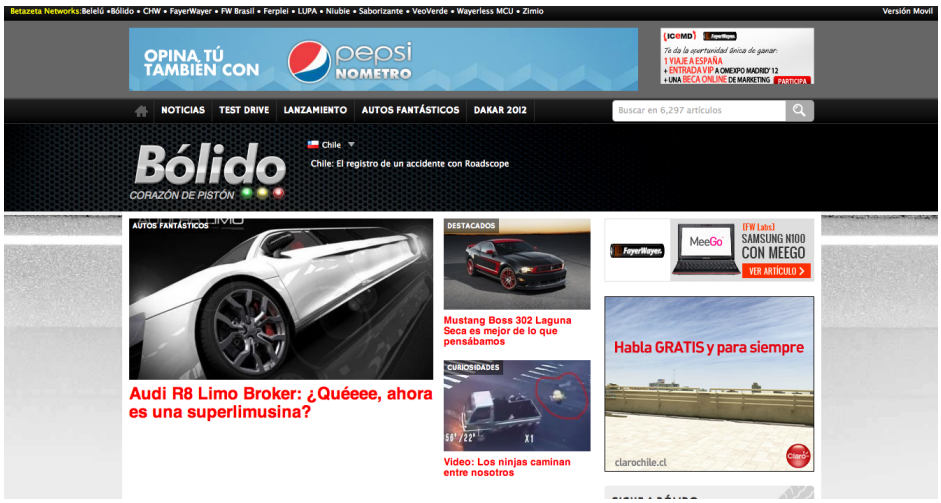


**Figura 3.5:** Blog Bólido.

**Fuente:** www.bolido.com.

> "Un sitio para los aficionados a los automóviles en general. Cubre deportes tuerca, así como temas mecánicos y estéticos, noticias y novedades, pruebas de manejo, consejos para conducir y valiosos datos técnicos y financieros para tomar la mejor decisión a la hora de comprar un vehículo".

La segmentación de contenidos en la línea editorial de Bólido actualmente es: Noticias, Test Drive, Lanzamiento, Autos Fantásticos y Dakar 2012.

Algunos de los titulares de publicaciones en este blog son:

- "Audi R8 Limo Broker: ¿Qué, ahora es una superlimusina?"
- "Nissa adelanta el GT-R Nismo GT3"
- "Volkswagen presenta el Up! de 5 puertas"
- "Policía de Berlín recibe 3 Toyota Prius"

### 3.9.3. FayerWayer

FayerWayer es un sitio que habla sobre ciencia y tecnología, ver figura 3.6. De acuerdo a la descripción de betazeta.com:

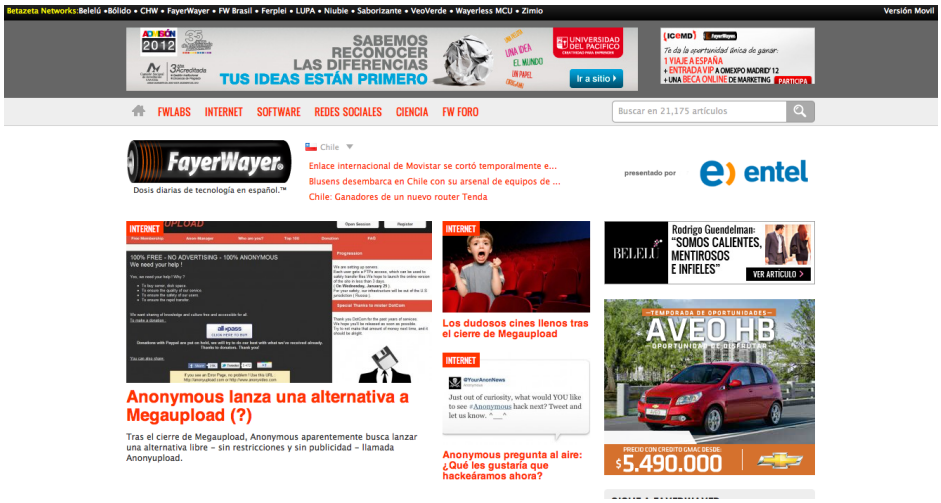


**Figura 3.6:** Blog FayerWayer.

**Fuente:** www.fayerwayer.com.

> "FayerWayer es el blog de tecnología más leído de América. ¿Nuestro plus? No solo entregamos información si no que también generamos discusión y conversación sobre un mundo donde el futuro es el protagonista".

La segmentación de contenidos en la línea editorial de FayerWayer actualmente es: FW-Labs, Internet, Software, Redes Sociales y Ciencia.

Algunos titulares que se pueden encontrar en este blog son:

- "Anonymous lanza una alternativa a Megaupload"
- "Partido Pirata catalán unifica denuncias de afectados por el cierre de Megaupload"
- "Mark Zuckerberg, artista del grafiti"
- "Científico del MIT anuncia procesador con 100 núcleos"

### 3.9.4. Ferplei

Ferplei es un blog dedicado al fútbol, ver figura 3.7. La descripción del blog dice lo siguiente:


**Figura 3.7:** Blog Ferplei.

**Fuente:** www.ferplei.com.

> "Ferplei es una comunidad ligada al fútbol que busca informar y entretener a los fanáticos del deporte rey. Goles, jugadas, noticias al minuto y un revolucionario método de transmisión de los partidos vía twitter son parte de su menú informativo".

La segmentación de contenidos en la línea editorial de Ferplei actualmente es: Liga Chilena, Liga BBVA, Cartelera, Guapas (Segmento con fotografías de modelos) y Chilenos en el extranjero.

Algunos de sus titulares:

- "Blatter anunció que en Brasil 2014 no se permitirán más goles "fantasmas" con ayuda de la tecnología"
- "Braulio Leal ante choques con Tigres: "Queremos ser protagonistas estos dos partidos""
- "Video: Túnez ganó gracias a Khalifa y el anfitrión Gabón comenzó celebrando"
- "Villarreal logró el triunfo que lo sacó la zona de descenso en la Liga BBVA"

### 3.9.5. Niubie

Niubie es un blog dedicado a los videojuegos, ver figura 3.8. En betazeta.com es posible encontrar la siguiente descripción:


**Figura 3.8:** Blog Nuibie.

**Fuente:** www.niubie.com.

> "Una comunidad de videojuegos, para gamers de todas las edades, colores, perfiles e interés. Todo lo que se pueda jugar con una consola está en este lugar. Críticas, noticias, avances, concursos son parte del mundo niubie".

La segmentación de contenidos en la línea editorial de Niubie actualmente es: Niubie Labs, PC, PS3, XBOX 360, Playstation Vita, iPhone, Nintendo 3DS, PSP y Wii.

Algunos de sus titulares:

- "Estos son los precios y las fechas del Xbox Live House Party 2012"
- "Futurología: Anunciarían God of War IV y Syphon Filter IV durante febrero próximo"
- "Street Fighter X Tekken no contará con personajes exclusivos en Xbox 360"
- "Las citas en "Love Plus" serán más realistas con el Nintendo 3DS"

### 3.9.6. Saborizante

Éste es un blog dedicado a la cultura y panoramas, ver figura 3.9. La descripción de este blog dice lo siguiente:


**Figura 3.9:** Blog Saborizante.

**Fuente:** www.saborizante.com..

> "Saborizante es la cartelera cultural y de entretenimiento de Santiago. En ella se publicitan los eventos diurnos y nocturnos que llenan la agenda de sus lectores, además de regalarles entradas y premios a los integrantes más activos de la comunidad".

La segmentación de contenidos en la línea editorial de Saborizante actualmente es: Panoramas, Música, Fiestas, Conciertos, Teatro, Comer y Beber y Concursos.

Algunos de sus titulares:

- "Fueron felices para siempre: Esperanza termina con alto rating"
- "Mira la nueva -y sexy- campaña de Huntcha.com"
- "La Guacha gana con gran polémica el Festival de Olmué"
- "Rita Lee anuncia su retiro de los escenarios por problemas de salud"

### 3.9.7. VeoVerde

Veoverde es un blog dedicado al medioambiente, ver figura 3.10. La descripción del blog es la siguiente:


**Figura 3.10:** Blog Veoverde.

**Fuente:** www.veoverde.com.

> "VeoVerde es una comunidad que tiene como fin informar, conversar y discutir sobre la sustentabilidad y las diversas formas para respetar al planeta tierra y a nosotros mismos. Queremos invitarte a tomar conciencia y a informarte".

La segmentación de contenidos en la línea editorial de Veoverde actualmente es: Actualidad, Energía y Recursos, Flora y Fauna, Noticias, Ciencia y Ecología, Ecogadgets.

Algunos de sus titulares:

- "Llamarada solar que se avecina provocaría espectacular aurora boreal"
- "Como si de verdad el paraíso se hubiese construido en seis días"
- "Los delfines y las ballenas pueden ser grandes amigos"
- "Chile: Nacimiento de Hipopótamo pigmeo en el Buin Zoo es un hito a nivel mundial"

### 3.9.8. Wayerless

Wayerless se especializa en dispositivos móviles, ver figura 3.11. La descripción de este blog es la siguiente:


**Figura 3.11:** Blog Wayerless.

**Fuente:** www.wayerless.com.

> "Wayerless es el hermano móvil de FayerWayer. Busca satisfacer la creciente demanda que hay por información, no solo de los aparatos en sí, sino de la industria en general (accesorios, operadores, software y trucos) y también por el creciente negocio de los tablets".

La segmentación de contenidos en la línea editorial de Wayerless actualmente es: W Labs, iPhone, Android, BlackBerry, Nokia, Microsoft y Galaxy Note.

Algunos de sus titulares:

- "Los nuevos sensores CMOS de Sony mejoran las prestaciones de los smartphones"
- "29 % de los estadounidenses tiene una tableta o un e-reader"
- "5 motivos por los que RIM ha perdido su protagonismo"
- "Windows Phone Marketplace rebasó las 60 mil aplicaciones"

# Capítulo 4

# Solución Propuesta

En esta sección se describe el proceso de desarrollo y especificación de la solución propuesta. En primer lugar se mencionan algunas consideraciones respecto a la problemática a resolver, luego se procede a argumentar la selección de tecnologías, luego se habla de la arquitectura utilizada en Betazeta y la implementada por la aplicación, y finalmente se entra de lleno a la especificación del software, que incluye la descripción del modelo de datos, objetos que lo componen, sus funcionalidades principales y utilización.

Es importante destacar que al seguir un desarrollo basado en metodologías ágiles, el que sigue un proceso iterativo de diseño, implementación, pruebas y análisis, muchas desiciones en el diseño final del software se basan en pruebas experimentales. Por este motivo, al explicar el diseño del software es necesario hacer referencia a experimentos que permiten entender el por qué de las estructuras propuestas.

## 4.1. Consideraciones con tildes en el idioma Español

Antes que todo, dentro de las complicaciones que representa el idioma español en el área de análisis de texto, se encuentra el uso de tildes en las palabras.

Los tildes en un texto permite diferenciar algunas palabras de stop words, por ejemplo, *dé* (verbo dar) y *de* (preposición) respectivamente. Otra ventaja es que permite distinguir algunos tiempos verbales, como *amara* y *amará*. Finalmente nos permite desambiguar palabras en las que el tilde cambia el significado de ésta, como *médico* y *medicó*.

Por otro lado, de acuerdo al *Estudio de la incidencia del conocimiento lingüstico en los sistemas de recuperación de la información para el español*[11], se tiene que este tipo de palabras no ocurre comúnmente, por lo que es posible considerarlos como casos particulares del lenguaje. Sumado a lo anterior existe la problemática de que resulta imposible asegurar que una palabra, que proviene de una fuente desconocida, esté correctamente escrita, lo que

podría llevar al método a un error generando ambigüedades.

A pesar de que las ventajas de los tildes permiten enriquecer el análisis semántico, la cantidad de palabras que podrían diferenciarse no son representativas respecto del volumen de texto a analizar. Por otra parte, el costo que significa incluir esta diferenciación es alto, puesto que aumenta el procesamiento requerido e incrementa el tamaño del diccionario a evaluar. Por lo anterior, en este proyecto se ignorarán los acentos en las palabras y se realizará un proceso de normalización en todo el texto estudiado.

## 4.2. Selección de tecnologías

### 4.2.1. Selección del lenguaje y framework de implementación

Según lo anterior, una alternativa acorde a las necesidades del cliente es desarrollar un sistema web que permita integrarse a los sistemas que utiliza la empresa. Ahora bien, se realizó una breve investigación en la web respecto a qué lenguajes podían resultar convenientes para ésta tarea. En el sitio web de David M. Blei[1], creador del modelo LDA analizado en este trabajo, es posible encontrar diversas implementaciones de dicho método en lenguajes como C, C++, Python y R. Por otro lado, también fue posible encontrar implementaciones del algoritmo en Ruby[2]. PHP es el lenguaje utilizado para el desarrollo en Betazeta, dado que Wordpress lo utiliza, y encontrar desarrolladores en este lenguaje es relativamente simple, pero presenta la limitación de no poseer recursos dentro del área de text-mining, por consiguiente, evitando "re-inventar la rueda" se descartó como lenguaje para el desarrollo de ésta plataforma.

Ruby y Python son lenguajes que permiten hacer una aplicación web en forma simple entregando poderosas herramientas para ello, utilizando una sintaxis poco verbosa. En la actualidad, ambos son populares, teniendo amplio soporte por parte de una comunidad abierta de desarrolladores además de que los dos poseen robustos frameworks de trabajo.

En el caso de Ruby, Ruby on Rails es incuestionablemente el framework más popular para este lenguaje. Por otra parte, en el sitio web de Python[3] es posible encontrar un listado de los *frameworks* disponibles para el lenguaje, donde los más relevantes en función de la comunidad de usuarios que los respalda son: Django, Zope, Pylons y Web2py. En la figura 4.1 es posible observar la cantidad de búsquedas de Google asociadas a cada uno de los frameworks, lo que revela la participación de la comunidad de desarrolladores sobre cada una de las plataformás. De acuerdo a estos datos se observa que Django es el framework que recibe más consultas en el buscador Google, por ende es posible inferir que posee mayor respaldo dentro de la comunidad de desarrolladores.

[1] http://www.cs.princeton.edu/ blei/topicmodeling.html
[2] https://github.com/ealdent/lda-ruby/wiki/
[3] http://wiki.python.org/moin/WebFrameworks

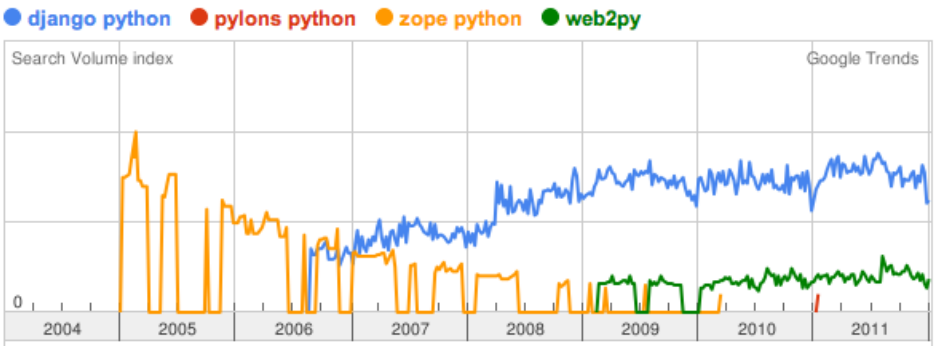


**Figura 4.1:** Impacto en Google Trends de los frameworks más relevantes sugeridos por Python.org.

**Fuente:** Google Trends.

En la figura 4.2 es posible observar una comparativa de las búsquedas de Google entre los dos principales frameworks de Python y Ruby, en este gráfico se observa la sostenida popularidad de Django mientras que Ruby presenta una abrupta caída en el útlimo tiempo. En este marco se eligió Python por sobre Ruby por presentar más recursos en el área de Topic Models, además de tener conocimientos previos en este lenguaje, así como su framework.

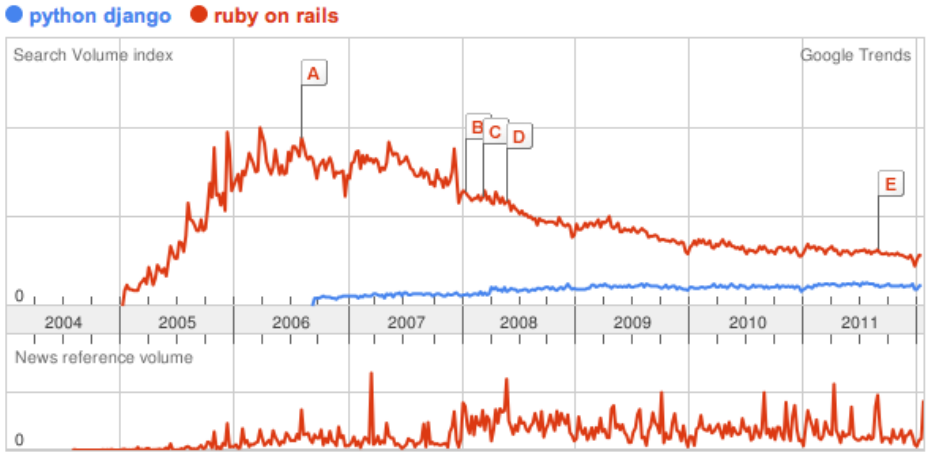


**Figura 4.2:** Impacto en Google Trends entre los frameworks Ruby on Rails y Django para Python.

**Fuente:** Google Trends.

### 4.2.2. Selección del motor de base de datos

En este caso se optó por trabajar con MySQL. Por un lado este motor es libre, ampliamente utilizado en el mercado y es utilizado por WordPress. Además de lo anterior, Betazeta posee la mayoría de sus proyectos funcionando sobre este motor.

Además, MySql permite trabajar con altos volúmenes de datos. Más aún, en el caso de la aplicación, el conjunto de textos a analizar siempre estará acotado, esto se debe a que el análisis de texto en el marco de la empresa siempre está asociado a una componente temporal. Lo anterior implica que pierde sentido hacer un análisis de todos los post existentes en la historia de los blogs, puesto que el contenido tiende a quedar obsoleto en el tiempo. Incorporar estos datos que dejan de ser relevantes en un modelo de clasificación genera ruido. Este fenómeno se observa de forma aún más acentuada en blogs asociados a tecnología, pero en general aplica a todo tipo de temas cuyo contenido evolucione en el tiempo.

Por otro lado, en caso que MySQL impida escalar, ya sea porque el volumen de datos supere con creces las expectativas o la empresa decida migrar sus aplicaciones a otro tipo de motor, por ejemplo, un modelo noSQL como es el caso de MongoDB, la utilización de un ORM, proporcionado por el framework Django, en diversos módulos de la aplicación simplifican el proceso de migración, ya que sólo es necesario especificar el tipo de base de datos y Django se ocupa del resto y su utilización es absolutamente transparente. Por motivos de eficiencia el ORM no puede ser utilizado en los componentes críticos de la aplicación y por tanto es necesario realizar consultas explícitas a la base de datos, debido a lo anterior, este set de consultas se encuentran acotados y bien definidas por lo que la migración, en caso de ser necesaria, no debiese ser un problema.

## 4.3. La Arquitectura

En esta sección se describen los componentes que conforman la aplicación y su interacción. En primer lugar se exponen las principales razones que motivan a elegir la arquitectura propuesta, para luego describir a grandes rasgos los distintos módulos del software y cómo interactúan entre sí.

### 4.3.1. La arquitectura de Betazeta

Al comienzo del proceso de investigación de este trabajo se aprendió todo lo que involucra la problemática de Betazeta, desde el proceso de redacción con el que se generan los documentos hasta la plataforma tecnológica utilizada para publicar contenido.

Betazeta actualmente trabaja con Wordpress[4] el cual está integrado en toda la red de blogs de Betazeta como herramienta de publicación de sus posts. Se utiliza una única configuración

[4] http://es.wordpress.org/

de ésta plataforma para todos los sitios, sólo algunas componentes del aspecto gráfico son individuales entre blogs, como colores y logos, pero incluso la estructura visual es la misma. Ver Figura 4.3.

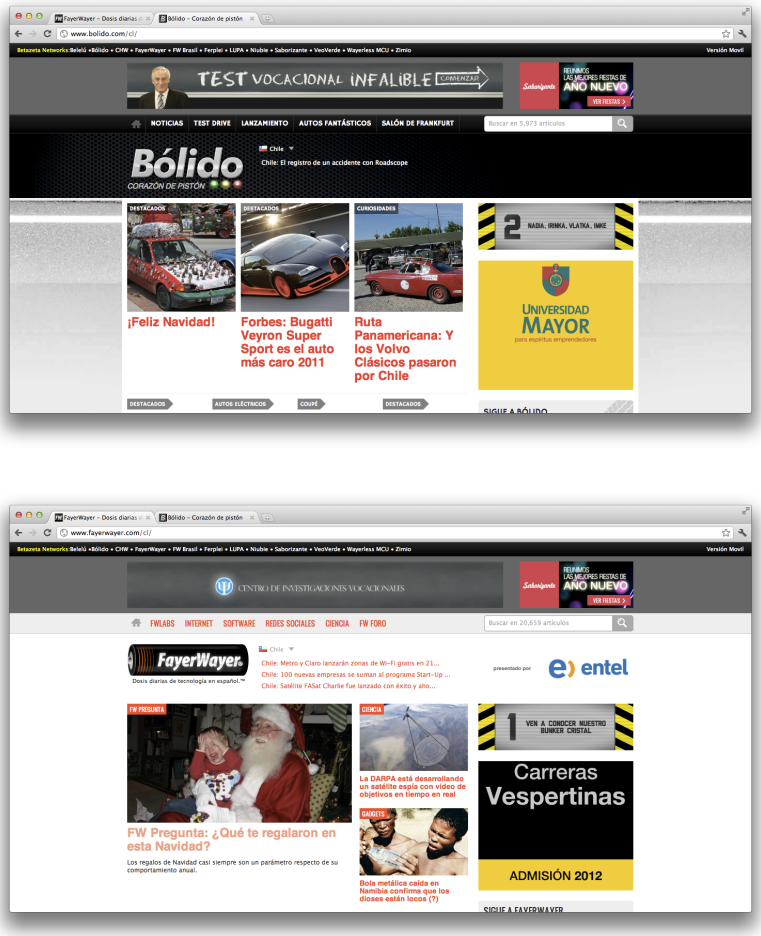


**Figura 4.3:** Diagramación del sitio Bólido (arriba) y FayerWayer (abajo) utilizando Wordpress. Es posible observar evidentes similitudes en las estructuras de ambos sitios.

**Fuente:** www.bolido.com y www.fayerwayer.com

La arquitectura de Betazeta mezcla el servicio de Amazon EC2 con máquinas locales Unix y permite soportar más de 8 millones de visitas mensualmente con peaks de uso que pueden llegar incluso a las 200.000 visitas en un sólo día. Por motivos de confidencialidad con la empresa esta arquitectura no puede ser descrita en detalle, pero se puede mencionar sus componentes principales y una breve descripción.

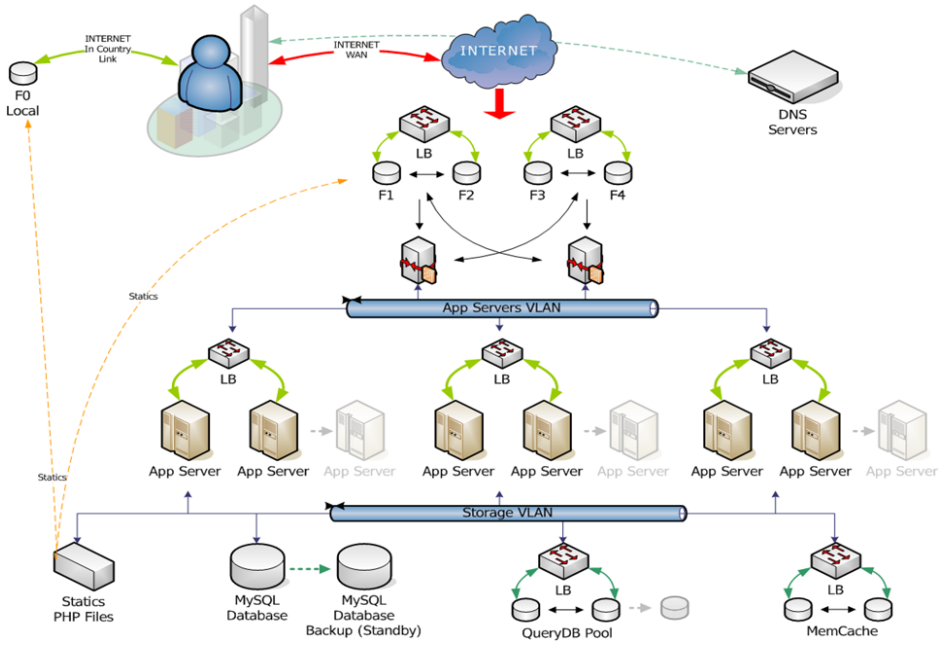


**Figura 4.4:** Arquitectura de los sistemas web en Betazeta Networks.

**Fuente:** Rodrigo Bustos, encargado de sistemas, Betazeta Networks.

Betazeta utiliza un sistema de múltiples niveles que permiten acceder a los datos en forma eficiente, ver figura 4.4. En el diagrama *LB* representa un balanceador de carga (*load balancer*). Cada vez que un contenido nuevo es accedido inevitablemente es necesario realizar un gran número de procesos y consultas a la base de datos, cuando esto ocurre, cada acción es almacenada en caché de manera que cuando se requiere nuevamente, se pueda acceder a la información inmediatamente.

Cada consulta a los sitios de Betazeta puede acceder a tres niveles distintos. En la primera capa, que tiene acceso directo a internet, se almacena el código HTML de una página, por ende cuando se debe enviar contenido que fue previamente accedido y ya se encuentra en caché, $F1$, $F2$, $F3$ y $F4$ en la figura, desplegar estos archivos es instantáneo y no es necesario ejecutar los procesos que generan dicho código. Cuando un sitio recibe algún tipo de actualización, el caché de este contenido debe ser elimnado, por lo tanto es necesario hacer una consulta en el sistema. Dicha consulta pasa a la capa de *App Servers VLAN* que realizan la petición requerida por el cliente. En este nivel se ejecutan procesos en lenguajes como PHP, Ruby o Node.JS desplegando el contenido requerido por el usuario y almacenándolo en caché. Cuando un proceso se ejecuta en App Servers VLAN usualmente requiere acceder a bases de datos, por lo que la consulta accede al tercer nivel. En el tercer nivel se tiene a disposición un conjunto de servidores que poseen un caché de consultas SQL. Si la consulta SQL se encuentra en caché entonces la respuesta se realiza de inmediato. En caso contrario se realiza una petición

al motor de base de datos y la consulta es ejecutada almacenado su resultado en caché para agilizar un segundo acceso.

Toda capa en la arquitectura posee un sistema de balanceadores de carga que permiten administrar eficientemente el uso de recursos.

De acuerdo a lo antes expuesto en esta sub-sección, la aplicación a desarrollar debe poder integrarse fácilmente en la infraestructura descrita. Esto es posible mediante un sistema web que se comunique con la plataforma de Wordpress y se pueda correr dentro de Amazon EC2. Lo anterior presenta múltiples ventajas que se derivan de poseer un sistema web, puesto que permite acceder al servicio en forma remota y concurrente. Además se puede aumentar la capacidad de procesamiento y memoria de los servidores virtuales de EC2, lo que permite mejorar el rendimiento y capacidad de la aplicación en forma dinámica según se requiera.

### 4.3.2. Arquitectura de la aplicación

Como ya se mencionó, la aplicación, denominada Tanalyzer (por análisis de texto en inglés), es una plataforma para la clasificación de texto aplicado a los post de los blogs de Betazeta y posee diversos módulos encargados de distintos procesos, todo contenido dentro del marco del framework Django.

Cada uno de los módulos posee un conjunto de modelos y vistas. Cada modelo representa una entidad o tabla en la base de datos, mientras que las vistas realizan la lógica del módulo. Un módulo en la jerga de Django se denomina *aplicación*. Cada aplicación debiese ser autocontenida cumpliendo un rol específico.

Tanalyzer posee una arquitectura SOA o Service Oriented Arquitecture, esto significa que cada aplicación provee un servicio disponible mediante una url. de ésta forma cada aplicación se mantiene independiente, realizando su función concreta, la que es posible utilizar en forma remota. Si el servicio requiere parámetros especiales de configuración se provee una interfaz para la recepción de variables mediante GET, esto implica incluir la variable en la url de acuerdo a algún patrón o bien por POST, donde las variables se envían mediante una petición HTTPRequest y no son visibles a simple vista. POST se utiliza, en este caso, para proporcionar las variables a métodos que requieren muchos parámetros.

Todas las funciones del software poseen esta arquitectura, por consiguiente es posible que la empresa utilice estos servicios para controlar la aplicación en forma externa, sin embargo, una de las aplicaciones corresponde a una interfaz web donde es posible administrar todos los procesos y es la encargada de realizar las llamadas a los otros servicios disponibles. Esta interfaz está pensada para poder simplificar el proceso de captura y manipulación de documentos, además de poder visualizar los datos relevantes y comprender cómo funciona un modelo particular.

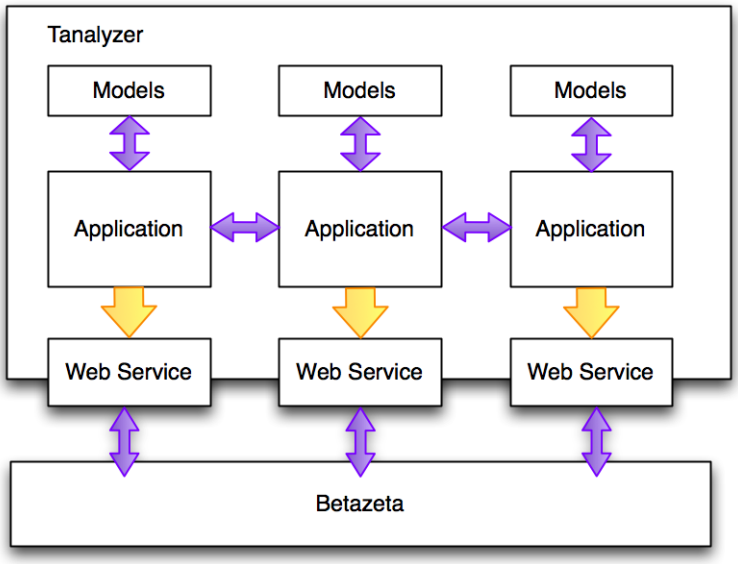


**Figura 4.5:** Arquitecutra de Tanalyzer.

**Fuente:** Elaboración propia.

En la figura 4.5 se puede observar un diagrama de la arquitectura de Tanalyzer. En la imagen se observa cómo cada aplicación posee un modelo de datos asociado, su propio web-service y además la lógica de una aplicación puede interactuar con otras. Esto en particular es relevante, ya que algunos modelos de datos se usan en múltiples aplicaciones.

### 4.3.3. Implicancias de LDA en el diseño de la aplicación

Durante el inicio de este proyecto se seleccionó LDA como alternativa de modelo de clasificación en base a las referencias citadas en el marco conceptual. Se realizaron algunos experimentos para validar la factibilidad de llevar a cabo los desafíos propuestos con resultados positivos y además se observaron algunos comportamientos que se detallan a continuación y que tuvieron incidencia directa en el diseño de la aplicación completa.

#### Utilización del modelo LDA en la práctica

El resultado de un modelo utilizando LDA corresponde a dos matrices. La primera matriz entrega la relación entre el conjunto de palabras analizadas en los textos y una temática implícita encontrada por el algoritmo. La segunda matriz representa la relación entre los textos

| Keywords | Valor |
|---|---|
| android | 0.0126970423977 |
| aplicacion | 0.00852802308995 |
| iphone | 0.00664185722743 |
| aplicaciones | 0.00639639728641 |
| windows | 0.00621705194252 |
| movil | 0.00618969417819 |
| nokia | 0.0060088289585 |
| moviles | 0.00549891062902 |
| operativo | 0.00546927305101 |
| google | 0.00491223857186 |

**Cuadro 4.1:** Keywords y su valor asociado a un tópico en particular.

**Fuente:** Elaboración propia.

analizados y las temáticas. Si se toman las palabras más relacionadas con un tema es posible intuir la temática encontrada, a estas palabras más relevantes se les denominará *keywords* o *palabras clave*. En la tabla 4.1 es posible observar el listado de las diez keywords más representativas de un topic junto con su respectivo valor.

Cuando un tópico posee un conjunto de keywords que comparten una temática fácilmente identificable, entonces podemos decir que se encontró un topic con éxito. Por ejemplo, si dentro de las cinco keywords más relevantes para un topic se encuentran las palabras: *colo*, *pelota*, *equipo*, *fútbol*, *árbitro* es posible determinar que la temática es *Fútbol Chileno*. A cada topic se debiese asignar una etiqueta o *label* para poder facilitar la clasificación de documentos y la identificación de un topic. Además de las Keywords, un apoyo para designar un label representativo a un tópic es leer los documentos más asociados a dicha categoría. En la tabla 4.2 es posible ver las diez keywords y los diez documentos más relacionados a la temática cuyo label fue designado como Dispositivos Móviles.

Utilizando los valores pertenecientes a la matriz de palabra-tópico es posible tomar un texto no analizado y determinar un puntaje asociado a cada uno de los tópicos en base a las palabras de este nuevo texto, siempre y cuando posea palabras que ya fueron evaluadas por el modelo LDA.

Si bien utilizando la fórmula 3.8 es posible obtener una aproximación de cuán relacionados están el nuevo documento y un tema en particular, experimentalmente se observan casos en el que la distribución de palabras generan puntaje similar en dos textos diferentes, sin embago su distribución es completamente distinta. El caso más ejemplificador de ésta situación es tomar dos textos que poseen muchas palabras. Recordemos que para una palabra, su asociación a un tópic se mide con valores que van entre 0 y 1 con 1 implicando máxima asociación. En el primer documento se tiene que la mayoría de las palabras están asociadas al tema en estudio

| Topic: Dispositivos Móviles | |
|---|---|
| **Keywords** | **Titulares** |
| android | Adobe nos explica por qué finalizaron Flash para móviles *Wayerless* |
| aplicacion | España: Vodafone ofrece la Playbook para autónomos y empresas *Wayerless* |
| iphone | Swordfish es el nombre en clave de la próxima versión del navegador Opera *Fayerwayer* |
| aplicaciones | Windows Phone Marketplace alcanza las 50.000 aplicaciones publicadas *Wayerless* |
| windows | España: Llega Mi Mstore, un servicio multidispositivo de Movistar para gestionar aplicaciones *Wayerless* |
| movil | PRemoteDroid: Controla tu computador desde tu Android *Wayerless* |
| nokia | Turboactualizaciones: Ya está disponible el beta de Firefox 8 *Fayerwayer* |
| moviles | Microsoft da una primera vista a Windows 8 *Fayerwayer* |
| operativo | 'TV España App' lleva los canales locales y nacionales a tu Android o iOS *Wayerless* |
| google | En Wayerless: Los Imperdibles de la Semana *Wayerless* |

**Cuadro 4.2:** Keywords y artículos más representativos de un tópico.

**Fuente:** Elaboración propia.

por un valor de 0.03, mientras que otro documento posee sólo algunas palabras pero cada una de ellas tiene un peso de 0.6. Es importante distinguir que cuando se tiene una palabra con un valor de asociación de 0.03 se está hablando de una palabra más bien genérica que no tiene relación consistente con el tema, no así una palabra con un valor de 0.6. En el caso anterior, si los textos poseen los tamaños adecuados el puntaje obtenido por 3.8 podría ser similar, lo que nos lleva a inducir un error de falso positivo en el caso del texto que contiene muchas palabras vagamente relacionadas.

Para contrarrestar la situación anterior, se plantean dos mecanismos: el primero consiste en evitar las palabras poco relevantes en un topic y que simplemente por su volumen sumen un valor relevante, para ello se eliminan, por cada topic, todas las asociaciones de palabras que son menores a una cierta tolerancia y de ésta forma son ignoradas de la fórmula, pues aportan ruido al clasificador. El segundo mecanismo es una modificación a la fórmula 3.8, este cambio intenta introducir el concepto de palabras relevantes y no relevantes para una temática.

Una palabra es relevante para un topic cuando su valor de relación está por sobre una tolerancia previamente establecida. Por otro lado, las palabras no relevantes son aquellas que fueron removidas de la lista de asociación a la temática en particular como se mencionó en el párrafo anterior.

Intuitivamente, si un texto posee una alta tasa de palabras relevantes para un topic, es esperable que el texto esté relacionado con dicho tema; por otra parte si un texto posee muchas palabras no relevantes es esperable que dicho tema no pertenezca a esa temática, a pesar de tener palabras relevantes, ya que es posible que éstas estén siendo utilizadas en otro contexto semántico.

Lo anterior se traduce en la siguiente fórmula de clasificación:

$$p(d_i|c_j;\theta) = (\prod_{t=1}^{|V|} \frac{p(w_t|c_j;\theta)^{N_{i,t}}}{N_{i,t}!}) * \frac{RW_{i,j} * rw}{NRW_{i,j} * nrw} \tag{4.1}$$

Donde $RW_{i,j}$ es el número de palabras relevantes (Relevant Words) asociadas a la temática $j$ en el documento $i$, de la misma forma $NRW_{i,j}$ representa el número de palabras no relevantes (Not Relevant Words) asociadas a la temática $j$ en el documento $i$. $rw$ y $nrw$ son constantes que permiten ajustar la fórmula para determinar si se le da mayor relevancia al número de palabras no relevantes o al de las que sí lo son.

**Observaciones experimentales en el modelo LDA**

Al realizar experimentos con el agloritmo se observó que cuando LDA entrena textos de varios blogs simultáneamente, las temáticas resultantes están directamente relaciondas a la segmentación de Blogs sobre la que se entrenó. Dicho de otra forma, cuando se entrena LDA con textos de Belelú, Bólido, Ferplei, Niubie, Veoverde, Saborizante y FayerWayer, cada uno con un volumen similar, LDA encontrará 7 temas relevantes que tenderán a ser: Mujer, Autos, Fútbol, Video Juegos, Medio Ambiente, Panoramas y Tecnología, pero se dificulta encontrar temáticas implícitas más específicas dentro de estas categorías, dada la alta segmentación de los textos.

Cuando se entrena un modelo utilizando sólo los datos de un blog en particular, por ejemplo Fayerwayer, es posible encontrar temáticas asociadas a: Apple, Redes Sociales, Emprendimiento, Ciencia, etc. Es decir, si entrenamos sólo un conjunto de datos aislado por Blog, es posible modelar temáticas específicas asociadas a un área en particular.

Por otro lado, para Betazeta es importante poder determinar a qué blog o área pertenece un texto desconocido y además en qué categoría específica es posible clasificarlo. Dado el comportamiento de LDA se intuye que este problema se puede resolver. Por ejemplo si se tiene:

1. Un texto que habla sobre tecnología.
2. Un modelo que incorpora a todos los blogs.
3. Un modelo que posee sólo los textos de Fayerwayer y Wayerless.

En la situación anterior el primer modelo podría indicar que dicho texto es de tecnología y en consecuencia tiene alta probabilidad de pertenecer a FayerWayer o Wayerless, y el segundo modelo podría definir a qué categoría en particular se asocia el texto si se asume que ambos clasificadores son suficientemente precisos.

A partir del razonamiento anterior se diseñó la estructura de clasificación multimodelo que se describe a continuación. Para ahondar en los experimentos realizados y los resultados respectivos revisar el Capítulo 5.

### El sistema de clasifición

En la sección anterior se mencionaron las ideas que convergen en el diseño de un modelo jerárquico de clasificación. La intuición indica que un conjunto de modelos anidados permitirían filtrar y obtener clasificaciones con mayor granularidad sobre un nuevo documento. En base a esto se diseña el sistema de clasificación que se observa en la figura 4.6.

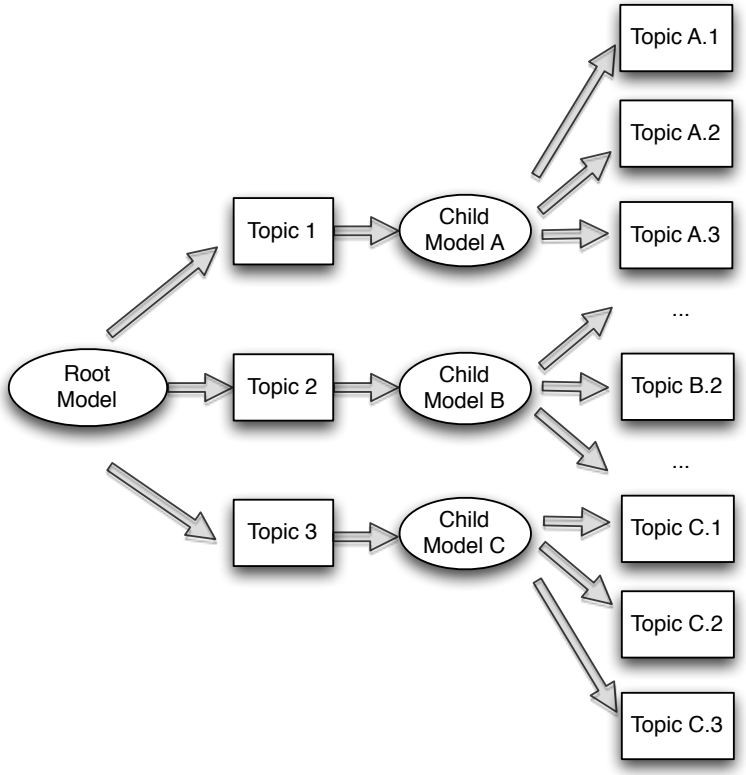


**Figura 4.6:** Sistema de clasificación multimodelo.

**Fuente:** Elaboración propia.

Un clasificador representa un árbol de modelos LDA. En la raíz del árbol se define el modelo inicial. Las hojas del árbol son representadas por los tópicos obtenidos en el modelo.

Luego cada tópico se conecta con un modelo LDA espécificamente entrenado en el área del texto y así sucesivamente.

Bajo este diseño, el modelo raíz siempre debiese actuar como filtro, incluyendo la mayor cantidad de información posible. En el caso de Betazeta, el modelo raíz debería ser entrenado utilizando todos los post de todos los blogs para generar un conjunto de topics genéricos por área. Por otro lado, las hojas del árbol deben ser modelos asociados directamente a los topics. En el caso de Betazeta, el modelo raíz si se entrena con todos los sitios, como ya se mencionó, se obtendrán topics del tipo: Autos, Tecnología, Mujer, etc. Por tanto, se debe entrenar un modelo por cada topic que represente una temática diferente. Para el caso del topic Tecnología se debería preparar un modelo que contenga sólo las publicaciones de Betazeta que toquen este tema tomando, por ejemplo, los post de FayerWayer y Wayerless o incluso FayerWayer, Wayerless y Niubie. El modelo resultante se debe enlazar al tópic Tecnología en dicho clasificador. El proceso se realiza para cada temática genérica encontrada. Es posible generar tantos niveles en el árbol como sea necesario, siempre y cuando se tenga textos suficientemente específicos, normalmente con dos niveles parece suficiente para obtener las clasificaciones generadas por las líneas editoriales de Betazeta (ver Capítulo 5).

### 4.3.4. Entidades

Para comprender la aplicación y los diferentes procesos, es necesario especificar qué entidades existen y qué rol cumplen dentro de la aplicación.

#### Client

Ésta entidad representa a un cliente. Cuando un modelo está entrenado y éste es asignado a un clasificador, es necesario definir quién tiene acceso a utilizarlo mediante el webservice de clasificación. Se asocia entonces un cliente a un clasificador, pudiendo existir más de un cliente asociados al mismo clasificador.

Betazeta se encarga de vender servicios relacionados a su plataforma, por lo que eventualmente podría ofrecer el webservice como un servicio de clasificación en base a sus blogs y por ello filtrar el acceso es relevante. En estos momentos existe un único cliente definido que es la propia empresa. Por otra parte, el software al ser genérico, puede ser utilizado por múltiples clientes que manipulen texto.

#### Dataset

Ésta es una de las entidades más relevantes del software. Un DataSet representa un conjunto de documentos. En el contexto de la empresa, un DataSet normalmente representa un Blog y cada documento es un Post, pero no es restrictivo. Ésta entidad posee campos asociados a un blog, como url y número de post por página en el inicio del sitio, que se utilizan

en uno de los procesos de obtención de documentos, pero para efectos del resto del proceso simplemente representa un set de datos genérico.

Al entrenar, la plataforma espera que los DataSets posean documentos relacionados a cierto tipo de contenido, en este caso los Post de un Blog cumplen este supuesto, pues todos los post están alineados al tema de dicho Blog y a una línea editorial. Por ejemplo, cuando se carga un DataSet que representa los Post de FayerWayer, es esperable que la temática central del DataSet sea la tecnología.

Cuando se quiere entrenar un conjunto de documentos en el que no se sabe nada sobre sus contenidos, lo mejor es asignar todos los documentos al miso Dataset.

#### Document

Ésta entidad representa documentos de texto. Como ya se mencionó, un Document puede representar un Post de un blog y estos se agrupan en Datasets. Al igual que Dataset, Document también posee campos directamente relacionados al blog de origen como número de comentarios y url del artículo, pero son sólo de referencia y no se utilizan en el proceso de entrenamiento, es por ello que la plataforma se mantiene como multi-propósito a la hora de agrupar y predecir nuevos documentos.

#### LdaModel

Representa un modelo de tópicos. Se llama LdaModel, puesto que la plataforma sólo utiliza el modelo Latent Dirichlet Allocation para generar los modelos. Se asocia con un conjunto de Datasets. A partir de los documentos contenidos en los DataSets se generará un modelo que representará la estructura de temáticas implícita asociada a estos documentos.

Un LdaModel puede ser entrenado utilizando sólo ciertos DataSets con el fin de generar modelos predictores específicos, como se mencionó anteriormente en la descripción de clasificadores.

#### Frequency

Se encarga de almacenar las frecuencias de palabras en cada documento. De acuerdo al modelo bag of words, en el que para un texto no importa el orden de sus palabras, ésta tabla del modelo de datos permite reconstruir los documentos.

**StopWord**

Como su nombre lo indica, ésta entidad representa una StopWord. Se utiliza en la fase de transformación o al ejecutar el método clean asociado a la entidad Document.

**Topic**

Ésta entidad representa una temática. Cada temática siempre está asociada a un LdaModel y posee un label para poder identificar de qué trata el topic.

**Word**

Éste objeto representa una palabra. Aquí se almacenan todas las palabras que componen los diferentes textos a excepción de las StopWords. A partir de este listado se puede saber si una palabra ya fue analizada en algún modelo, por consiguiente, si la palabra no está en esta tabla se sabe inmediatamente que no existe información asociada a ella como para poder hacer una predicción de un nuevo texto.

**Classifier**

Representa un clasificador. Al crear ésta entidad se define un LdaModel como la raíz del árbol de clasificación, como se especificó en la descripción del sistema jerárquico de modelos.

**ClassifierNode**

Representa un nodo en el árbol de clasificación. Cada entidad se asocia a un Classifier, uno de los Topic del modelo padre y el LdaModel que se utilizará con ese Topic.

**Otras entidades**

Además de las entidades mencionadas, se utilizan más entidades que realizan funciones específicas. Por ejemplo, *TopicWord* se utiliza para almacenar el valor de asociación entre un tema de un modelo particular y una palabra. *DocumentTopic* realiza la misma función pero con un documento. *DataSetLdaModel* asocia un conjunto de DataSets a un LdaModel. *DocumentDistribution* almacena una lista que representa un documento, esta lista contiene los ids de las palabras que lo componen y se utiliza como el input del algoritmo LDA. *WordDataSetFrequency* cuenta la frecuencia de cada palabra en un DataSet en particular y *WordLdaModel* almacena la lista de palabras que son relevantes en el conjunto de DataSets asociados a un LdaModel.

Además de estas entidades creadas en particular para la aplicación, existen entidades propias de algunos módulos de Django, dentro de las cuales se encuentra un módulo de usuarios disponible en forma nativa, que incluye un sistema de permisos, grupos y almacenamiento de sesiones, junto con otras componentes del framework. Es posible ver el modelo de datos completo en la que resta de este capítulo.

### 4.3.5. Diagramas de Clase

A continuación se presentan los diagramas de clase de la aplicación. Se destaca que bajo el estándar de Django, la lógica de la aplicación que se ejecuta en las vistas o views no son clases, si no que un conjunto de funciones, por ende no aparecen detalladas.

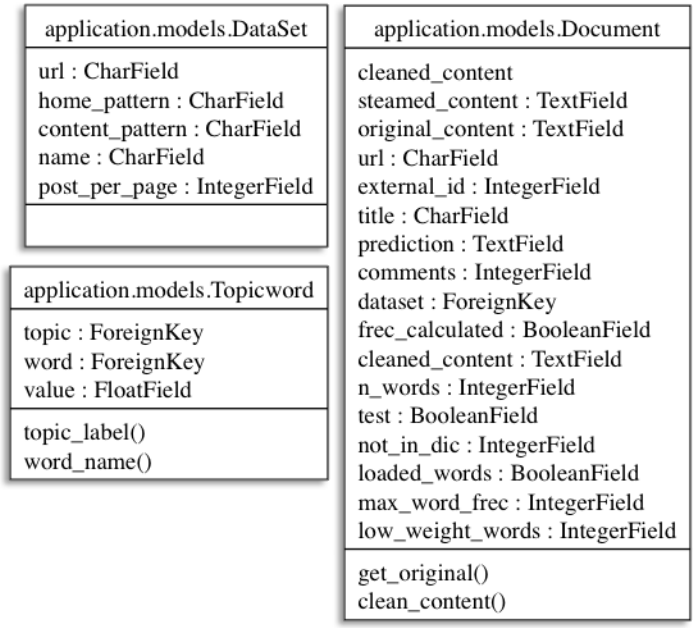


**Figura 4.7:** Diagrama de clases para Document, DataSet y TopicWord.

**Fuente:** Elaboración propia.

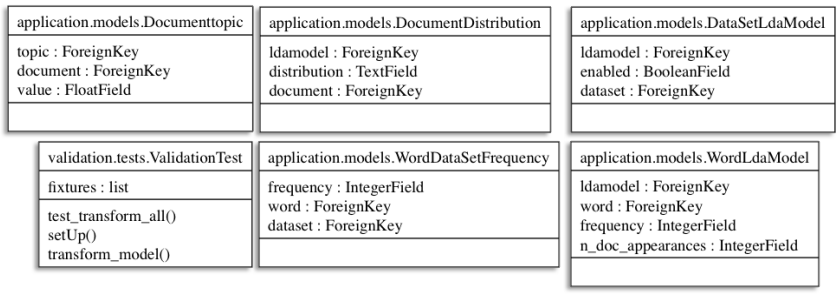


**Figura 4.8:** Diagrama de clases para DocumentTopic, DocumentDistribution, DataSetLdaModel, ValidationTest, WordDataSetFrequency y WordLdaModel.

**Fuente:** Elaboración propia.

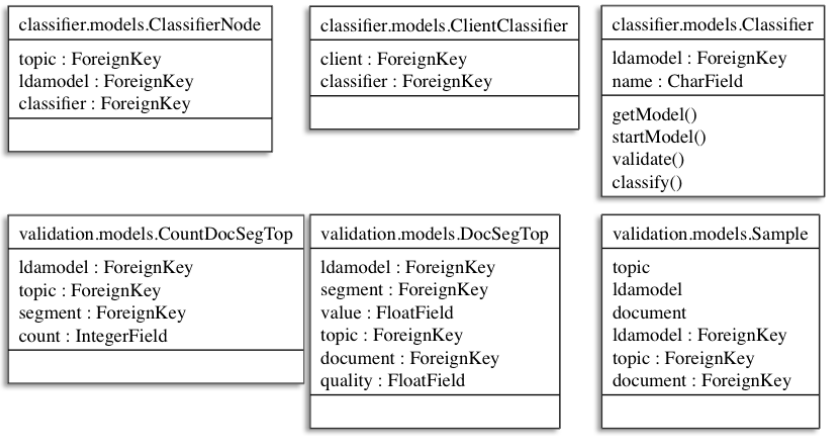


**Figura 4.9:** Diagrama de clases para ClassifierNode, ClientClasiffier, Classifier, CountDocSegTop, DocSegTop y Sample.

**Fuente:** Elaboración propia.

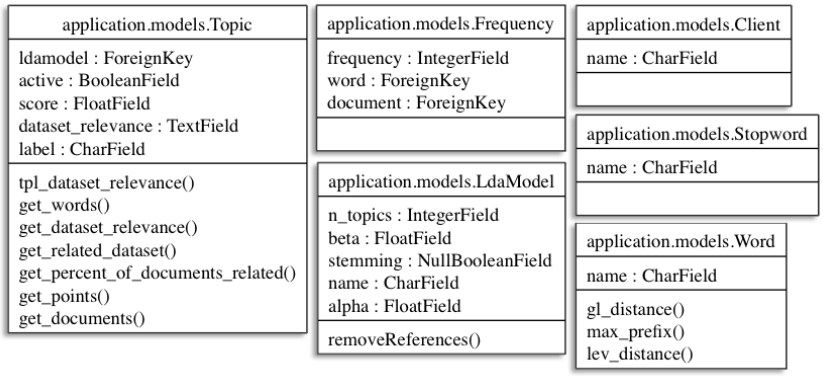


**Figura 4.10:** Diagrama de clases para Topic, Frequency, Client, LdaModel, Stopword y Word.

**Fuente:** Elaboración propia.

Además de lo anterior, la figura 4.11 muestra un diagrama simplificado de las relaciones de dependencia entre los distintos paquetes de la aplicación, esto incluye los modelos mencionados anteriormente, así como las vistas que contienen la lógica.

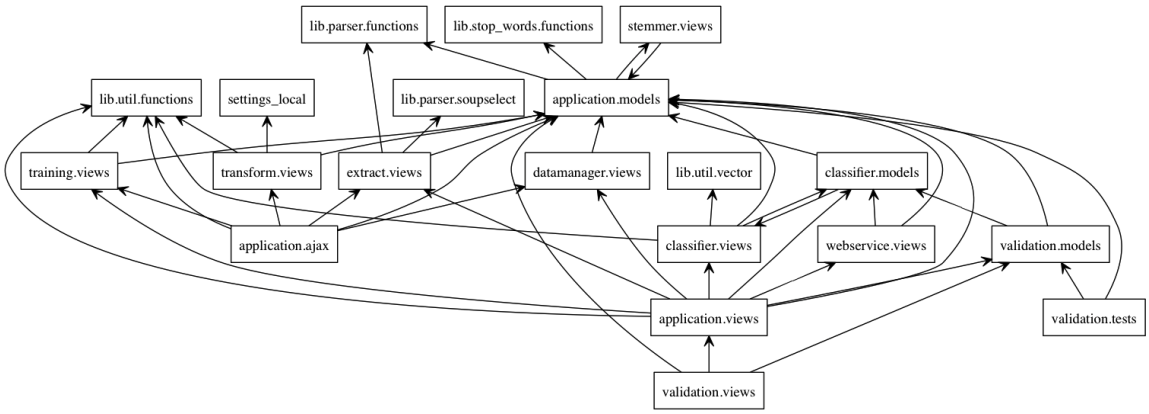


**Figura 4.11:** Relación entre los diferentes paquetes de la aplicación.

**Fuente:** Elaboración propia.

### 4.3.6. Modelo de Datos

De las entidades anteriores se desprende directamente el modelo de datos, el que se presenta a continuación. Debido a las múltiples relaciones que existen entre las entidades, estas se han agrupado de acuerdo a la función que cumplen. Las líneas punteadas, de acuerdo al éstandar, indican una relación de dependencia con otra entidad, las que muchas veces salen de la figura, indicando que existe una relación con una entidad que no se encuentra en el recuadro, pero basta con ver el nombre de las llaves foráneas para intuir las entidades relacionadas faltantes en cada figura, ya que se sigue la nomenclatura *nombreentidad_id* impuesta por Django.

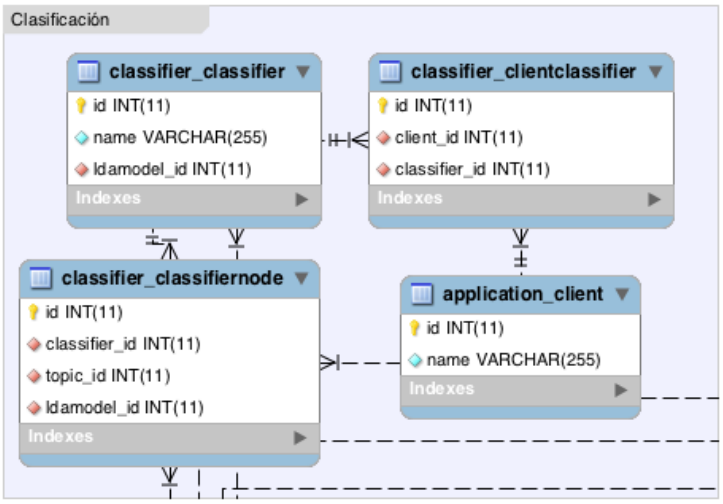


**Figura 4.12:** Diagrama de entidad relación para modelos Classifier, ClassifierNode, Client y ClientClassifier.

**Fuente:** Elaboración propia.

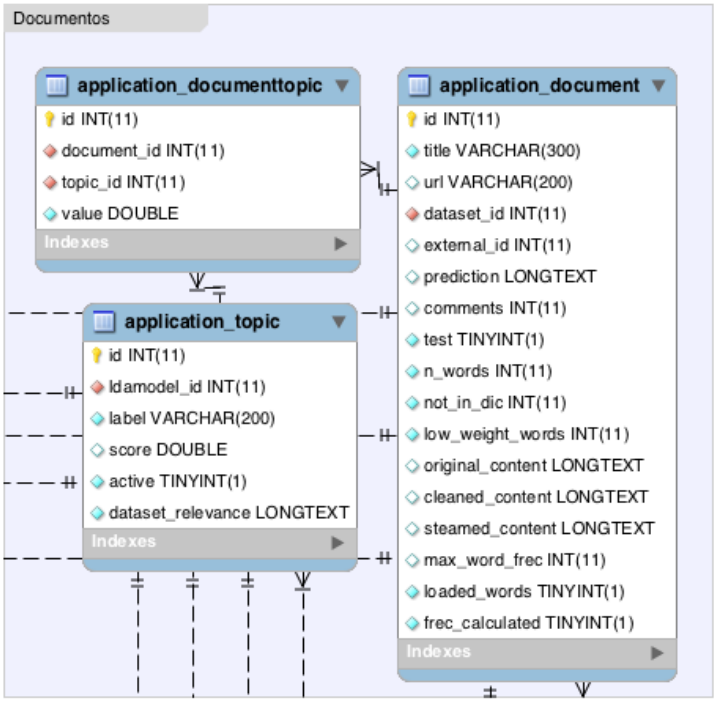


**Figura 4.13:** Diagrama entidad relación para modelos Document, Topic y Document-Topic.

**Fuente:** Elaboración propia.

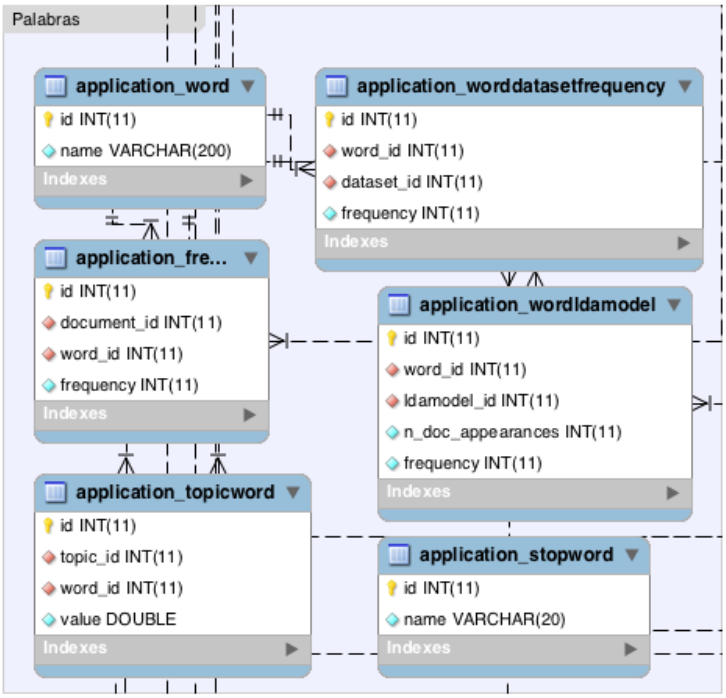


**Figura 4.14:** Diagrama entidad relación para modelos Word, Frequency, TopicWord, StopWord, WordLdaModel y WordDataSetFrequency.

**Fuente:** Elaboración propia.

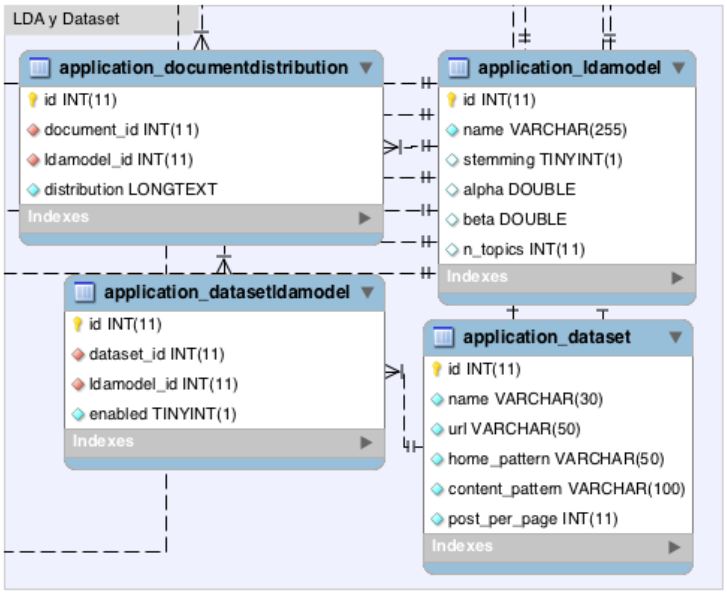


**Figura 4.15:** Diagrama entidad relación para modelos DataSet, LdaModel, DataSetLdaModel y DocumentDistribution.

**Fuente:** Elaboración propia.

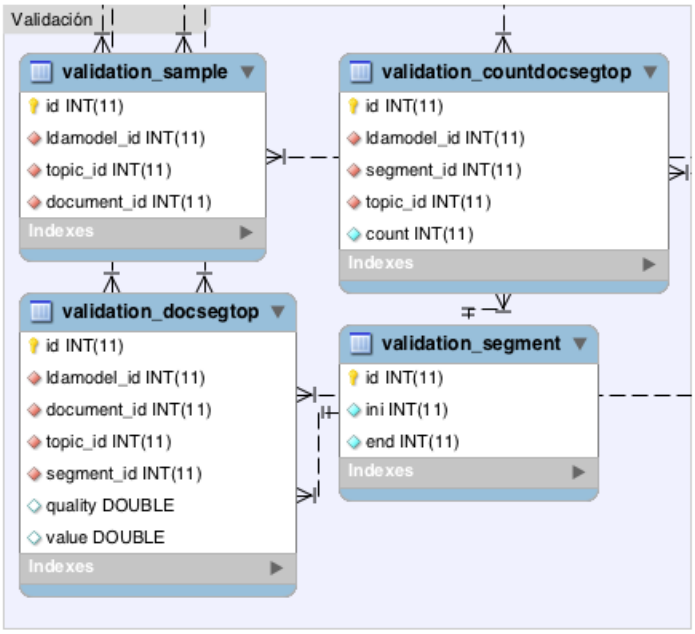


**Figura 4.16:** Diagrama entidad relación para modelos Sample, CountDocSegTop, Segment, DocSegTop.

**Fuente:** Elaboración propia.

### 4.3.7. Procesos

A continuación se describen a grandes rasgos las aplicaciones o módulos más relevantes dentro del software:

#### Extract

Esta componente del software obtiene los post para su análisis a partir de diversas fuentes usando dos mecanismos:

El primer método de extracción es hacer *Screen Scraping*, esta técnica consiste en leer el código fuente de un sitio web (su código *html*), y obtener información relevante en forma automática. Esto es una práctica común cuando un sitio presenta información de interés, pero no provee un mecanismo apropiado para distribuir su contenido más allá del código fuente en lugar de habilitar una API(Application Programming Interface).

Para obtener los datos se preparó un script que aplica esta técnica sobre los blogs conside-

rados en el estudio. Básicamente el script navega por un blog y lo recorre página a página. En cada una de las páginas detecta los links que apuntan al detalle de los artículos y obtiene los datos relevantes como título y contenido de una publicación. Éste proceso se hace generando hilos o threads con el fin de paralelizar la extracción de cada artículo acelerando la obtención de los textos. Dicha información se almacena en base de datos, en la entidad Document y se utiliza para poblar los algoritmos aplicados en etapas posteriores.

Este script se desarrolló utilizando Beautiful Soup[5], un analizador sintáctico de HTML/XML para Python que simplifica en gran medida esta tarea. Beautiful Soup transforma el código HTML en una estructura de árbol, lo que facilita la navegación y búsqueda de un elemento y una propiedad particular dentro del código fuente.

La principal motivación de utilizar ésta técnica es que a futuro es posible extender el método a otros sitios, lo que permitiría analizar blogs de la competencia, incluyéndolos dentro de la plataforma.

El segundo método es un script que permite realizar una conexión directa a una base de datos remota, obtener ciertos campos parametrizables de una tabla en particular que se ajustan para luego ser almacenados en el modelo de datos de Tanalyzer en la entidad Document, al igual que el proceso antes descrito. Este método es mucho más eficiente que realizar Screen Scraping, el único requisito consiste en tener acceso a la base de datos desde el servidor donde se aloja la aplicación.

Cada uno de estos scripts representan un método en la vista de la aplicación Extract. Para ejecutarlos la aplicación provee dos direcciones url, una para cada método a modo de Web Service. Cada url acepta ciertos parámetros vía POST.

### Transform

En esta fase se modifica un subconjunto de documentos, se aplican diversas transformaciones a los textos y palabras y se crean algunos datos para la utilización en etapas posteriores. En primer lugar se remueven caracteres no alfa-numéricos y se llevan todas las palabras a minúsculas. Luego se compara el texto con el listado de Stop Words y se remueven todas las apariciones de este tipo de palabras en todos los textos utilizando expresiones regulares.

Como ya se mencionó, una Stop Word corresponde a una palabra que frecuentemente aparece en múltiples textos independiente de su temática. Por esta razón no son relevantes para definir la naturaleza de dicho texto y por lo tanto es común removerlas para reducir la cantidad de términos a analizar.

Para llevar a cabo esta tarea se tomó dos conjuntos de Stop Words para español. El primero correspondiente al sitio web "El Webmáster"[6] y el segundo corresponde al proyecto

[5] http://www.crummy.com/software/BeautifulSoup/
[6] http://www.elwebmáster.com/referencia/stopwords-en-espanol

"SnowBall"[7]. Utilizando ambos textos se construyó una única lista, la cual se almacena en base de datos.

Opcionalmente se realiza Stemming sobre el texto para reducir el diccionario resultante asociado a las palabras y la dispersión de los conceptos que representa una familia de palabras. Por ejemplo, si la palabra caminar y sus derivados aparecen individualmente pocas veces podrían ser eliminados y no considerados en el análisis, mientras que la suma de todos los conceptos asociados a la acción de caminar, como caminaba, caminó, caminaron, etc. sí podrían tener alta relevancia, pero dada su segmentación no es posible reconocerla.

Luego de ésta fase se enumeran y almacenan todas las palabras obteniendo el número de veces que una palabra aparece en cada documento. Utilizando estos valores se eliminan del listado de palabras todas aquellas que en el conjunto completo de documentos posean baja frecuencia, esto es, palabras que tengan menos de un número determinado de apariciones en el total de textos, pues por su poca relevancia no agregan valor a los modelos resultantes, pero sí agregan un importante costo en el número de operaciones necesarias para todos los procesos que deben ser ejecutados posteriormente.

Además de la eliminación de palabras por baja frecuencia se procede a analizar la distribución de las palabras restantes en el conjunto de textos seleccionados para análisis. La aplicación asume que cada DataSet asociado al LdaModel bajo análisis está estrechamente relacionado a una macro-temática diferente de los demás DataSets, ya que usualmente se utilizan para representar un Blog por separado. Asumiendo este comportamiento, se analiza la frecuencia de apariciones de las palabras restantes por cada DataSet y cuando la frecuencia es relativamente constante es posible considerar esa palabra como una StopWord, por consiguiente también es eliminada del diccionario asociado al modelo en análisis. Es importante destacar que por cada modelo se define un sub-diccionario de palabras, este sub-diccionario de palabras es el resultado de los procesos antes mencionados y utilizando el diccionario particular de cada LdaModel se ejecuta el proceso de entrenamiento.

Cuando se han dejado sólo las palabras más representativas del set de textos se procede a llenar la tabla asociada a la entidad DocumentDistribution. Como ya se mencionó, ésta consiste en una lista con los ids de cada palabra.

### Training

Una vez que se termina el proceso de transformación opcionalmente se asigna la proporción de datos que se utilizará para validación y entrenamiento. Hecho lo anterior, es posible iniciar la fase de entrenamiento. En este caso el proceso Training recibe cuatro parámetros: $\alpha$, $\beta$, el número de temáticas a encontrar y el modelo a entrenar. $\alpha$ y $\beta$ son los parámetros que utiliza el modelo LDA para determinar las distribuciones iniciales de los documentos y sus palabras. $\alpha$ es la base para la distribución de documentos y $\beta$ para la distribución de palabras.

[7] http://snowball.tartarus.org/algorithms/spanish/stop.txt

En este proceso se obtienen todas las entidades DocumentDistribution asociados al modelo a entrenar, las que representan cada texto en forma de identificadores y se le entregan a la librería de LDA. Para la implementación de LDA se utilizó Delta LDA[2], extensión de LDA implementada por David Andrzejewski. Esta versión del algoritmo permite obtener set de temáticas asociadas a temáticas débiles en conjunto con los temas principales y es aplicado en la detección de erroes en el desarrollo de software, donde los temas débiles son las fallas en software, mientras que los temas fuertes están asociados al funcionamiento normal de un programa. Para ello se segmentan $\alpha$ y $\beta$ en dos componentes. Es posible utilizar DeltaLDA como LDA básico si se utilizan valores de $\alpha$ y $\beta$ constantes en lugar de segmentarlos. Ésta implementación se prefirió por sobre otras nativas en Python[8], puesto que está desarrollada en C e implementada en Python como un módulo, siendo rápido en su ejecución. El código fuente de este módulo está disponible en Internet[9]. El input de dicho módulo corresponde a los vectores $\alpha$ y $\beta$, un diccionario de python con un formato especial que representa todos los textos bajo análisis y el número de iteraciones que realizará LDA antes de entregar el resultado. Cada texto se representa como un diccionario compuesto de números, donde cada número representa una palabra del documento y se utiliza como identificador. Posee el requisito de que estos identificadores deben ser correlativos e ir desde 1 al número de palabras, para ello se hace una transformación que toma el id de la palabra de la base de datos y se asocia a un número del listado de identificadores correlativos generado temporalmente para el proceso de entrenamiento. Una vez terminado el proceso de entrenamiento se identifica la palabra utilizando su identificador temporal y se almacena con el id original que posee en la base de datos.

### Classify

Este módulo permite realizar clasificaciones sobre un texto utilizando las fórmulas antes mencionadas de Naïve Bayes 4.1 y los datos del modelo LDA previamente entrenado. Al llegar el texto se calcula el peso de cada una de sus palabras por cada topic y con ello se estima la probabilidad de pertenencia a las distintas temáticas. En ésta aplicación se definen las entidades de clasificación que componen el modelo jerárquico de tópicos.

### Webservice

Aquí se definen los diferentes webservice que estarán disponibles para realizar clasificación utilizano Tanalyzer. El webservice espera un texto mediante POST, el que es enviado al módulo Classify. Una vez recibida la respuesta de la componente de clasificación, se procesa de acuerdo a los datos definidos en el webservice y son enviados al cliente.

En la figura 4.17 se puede observar cómo interactuan los últimos 5 procesos con Betazeta a nivel general.

[8] http://www.mblondel.org/journal/2010/08/21/latent-dirichlet-allocation-in-python/
[9] http://pages.cs.wisc.edu/ andrzeje/research/delta_lda.html

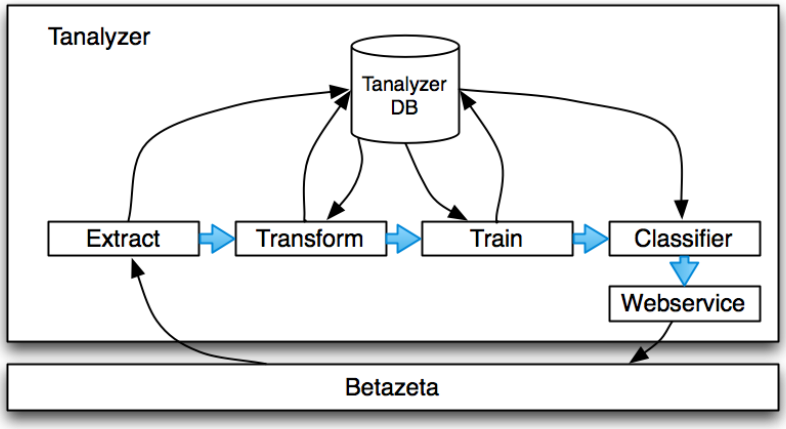


**Figura 4.17:** Arquitectura de la aplicación a nivel de procesos específicos.

**Fuente:** Elaboración propia.

### Validation

Validation es un módulo construido para obtener estadísticas de *Precision*, *Recall* y *F-Measure* en forma automática luego de clasificar manualmente un conjuto de textos a partir de las categorías generadas por un modelo. Estos resultados manuales se comparan con los valores entregados por el clasificador y a partir de esto se puede identificar qué tan bien se comportan las predicciones de un modelo.

### Application

Este componente provee la interfaz de usuario de la aplicación y conecta a todas las otras componentes independientes. Se conecta además con el sistema de usuarios de Django para proveer un método de autentificación al ingresar a la plataforma y su sistema automático de administración de modelos. En la próxima sección se describe este proceso en detalle.

## 4.4. Interfaz de usuario

Como ya se mencionó, Application consiste en la interfaz gráfica de la aplicación, a continuación se describe ésta componente utilizando imágenes que muestran cómo se visualiza cada componente.

### 4.4.1. Login y administración de entidades


**Figura 4.18:** Login del sistema Tanalyzer.

**Fuente:** Elaboración propia.

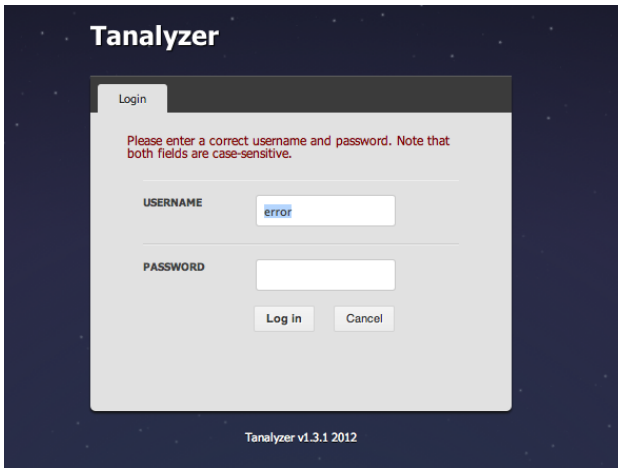


**Figura 4.19:** Error al ingresar datos en login.

**Fuente:** Elaboración propia.

Al iniciar la aplicación se encuentra la pantalla de login, ver figura 4.18. En ésta es necesario ingresar un usuario y password válido en el sistema. En caso de error, se notificará al usuario que los campos ingresados no son correctos 4.19.

Cuando el usuario se ha autentificado ingresa a la pantalla de inicio, ver figura 4.20. Aquí es

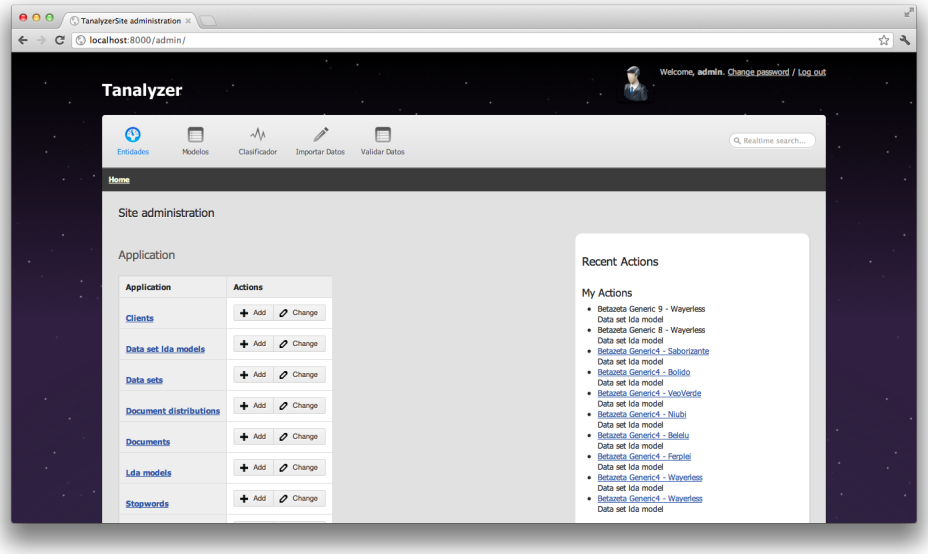


**Figura 4.20:** Pantalla de inicio.

**Fuente:** Elaboración propia.

posible visualizar el menú que se compone de las opciones: Entidades, Modelos, Clasificador, Importar Datos y Validar Datos. Por defecto el usuario ingresa al menú Entidades. Al costado derecho de ésta pantalla es posible visualizar un listado de las acciones recientes realizadas por el usuario. Al costado izquierdo se encuentra un listado de las diferentes entidades que existen en la aplicación y corresponden a las antes mencionadas como Client, DataSet, Word, LdaModel, etc. Cada entidad es administrable, esto quiere decir que se pueden crear nuevas entidades, visualizarlas (figura 4.21), modificarlas (figura 4.22) y eliminarlas, este conjunto de acciones se conoce como CRUD o Create, Read, Update y Delete en inglés.

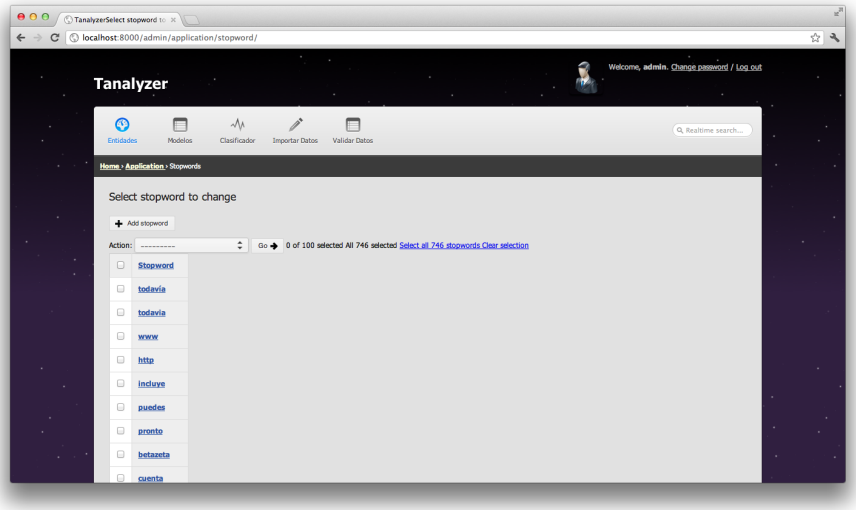


**Figura 4.21:** Listado de Stopwords en el sistema.

**Fuente:** Elaboración propia.

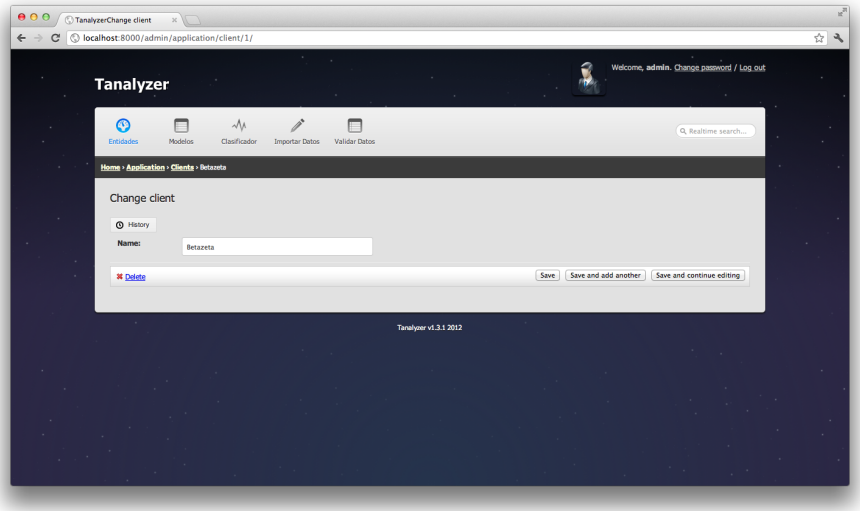


**Figura 4.22:** Edición de cliente.

**Fuente:** Elaboración propia.

En el encabezado del sitio se encuentra la información de la sesión del usuario, ver figura 4.23, donde es posible cambiar el password del usuario o cerrar la sesión. La administración de entidades y la gestión de usuarios es provista por Django y se genera automáticamente a partir de la definición de los modelos en cada aplicación.

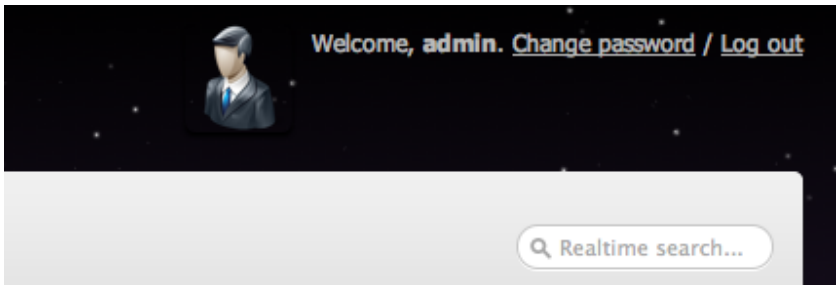


**Figura 4.23:** Cerrar sesión y cambiar contraseña.

**Fuente:** Elaboración propia.

### 4.4.2. Modelos

Al ingresar en la opción Models se presenta la interfaz de la figura 4.24. En ella es posible realizar los procesos de Transform y Train, seleccionar el tamaño del conjunto de datos de validación y mostrar el modelo resultante sobre un LdaModel en particular. Cada proceso sigue las especificaciones descritas en la sección anterior.

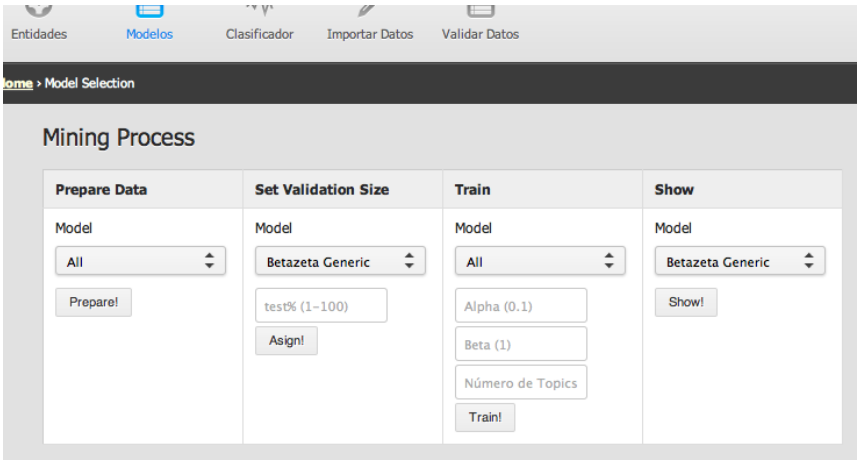


**Figura 4.24:** Interfaz que permite aplicar distintos procesos sobre un LdaModel.

**Fuente:** Elaboración propia.

### 4.4.3. Show

Cuando se selecciona la opción Show sobre un modelo, se muestra el listado de temáticas obtenidas después de entrenar dicho modelo. Aquí es posible:

- Modificar el nombre de los topics para asociarlos a conceptos concretos, haciendo doble click en su nombre.

Originalmente un nuevo topic recibe el nombre asociado al DataSet más representativo, esto significa que cuando se entrena un modelo usando los ocho blogs de Betazeta, los documentos más asociados al topic relacionado con Medio Ambiente serán los que pertenezcan a VeoVerde, por lo tanto a ese topic se le asigna la etiqueta *VeoVerde*, el nombre del DataSet, y posteriormente puede ser modificado por un nombre más específico como *Medio Ambiente*, ver figura 4.25.

- Ver la distribución dentro de los DataSets para un topic, ver figura 4.26
- Ver los keywords más relevantes de un topic, ver figura 4.27.
- Ver los documentos más asociados al topic, ver figura 4.31.
- Ver el porcentaje de documentos asociados al tópico, ver figura 4.30.

Lo anterior ayuda a definir un nombre o etiqueta correcta a cada temática. El porcentaje de documentos asociados se calcula contando la cantidad de documentos cuyo porcentaje de asociación con el tema es superior a un 15 % sobre el total de los documentos del modelo. Los porcentajes de asociación son obtenidos a partir del resultado directo del entrenamiento con LDA, en ese caso no se usa el clasificador diseñado en esta memoria. Se destaca que un documento puede estar asociado a más de un tópico, en consecuencia, la suma de los porcentajes de documentos asociados de todos los temas dará más de 100 %.

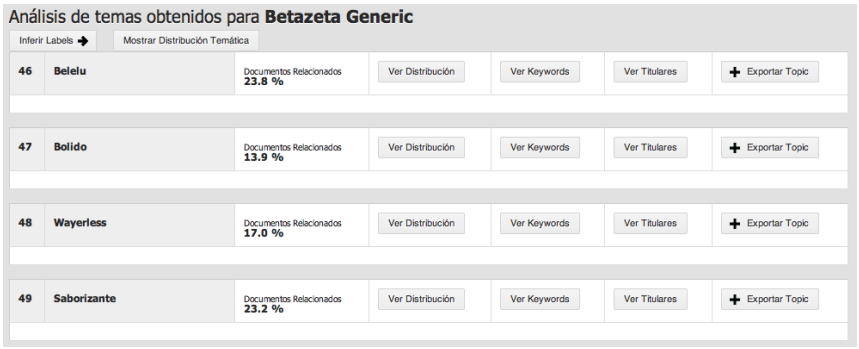


**Figura 4.25:** Listado de Topics encontrados después del proceso de entrenamiento.

**Fuente:** Elaboración propia.


**Figura 4.26:** Distribución para el topic Belelú.

**Fuente:** Elaboración propia.


**Figura 4.27:** Listado de Keywords más relacionadas al topic Bólido.

**Fuente:** Elaboración propia.

Una vez que se han asignado los nombres correspondientes a cada topic o clase obtenida es posible exportar las cien keywords más relevantes a un archivo de respaldo junto con el nombre o *label* asignado usando el boton Exportar, ver figura 4.28.

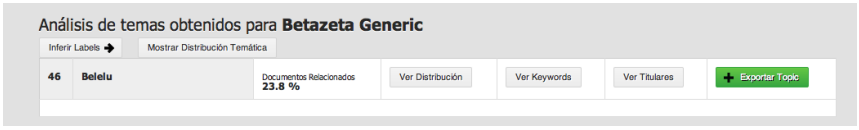


**Figura 4.28:** Exportar keywords de un topic.

**Fuente:** Elaboración propia.

Esto permite la utilización de la función de inferencia de nombres automática utilizando el botón *Inferir Labels* que aparece sobre el listado de temáticas, ver figura 4.29.


**Figura 4.29:** Inferir Labels.

**Fuente:** Elaboración propia.

Éste proceso interactúa con un archivo donde se almacenan los tópicos, en particular se guarda el nombre del tema y sus 100 keywords más relevantes.

- Se obtienen las 100 keywords más relevantes de los topics en el modelo.
- Se leen las 100 keywords almacenadas en el archivo de respaldo.
- Se comparan las keywords del modelo con las del archivo de respaldo.
- Si existe un 30 % de coincidencia entre las keywords de un topic del modelo y las de de un topic en el archivo de respaldo, entonces se define como nombre del tema el label almacenado en el archivo de respaldo para ese tópic.

Esto es útil cuando se prueban distintas configuraciones de modelos que trabajan sobre un conjunto común de datos o se espera encontrar topics que ya han aparecido en otros entrenamientos, así evitar el proceso de nombrar topics, pues consume un tiempo no despreciable puesto que es una tarea que se debe realizar manualmente.


**Figura 4.30:** Porcentaje de documentos relacionados al tópico Saborizante.

**Fuente:** Elaboración propia.

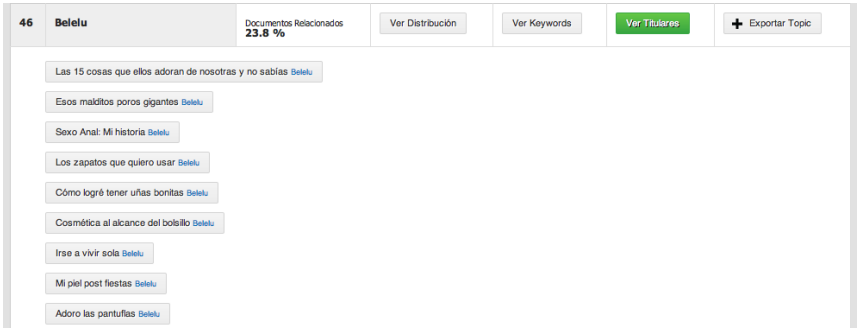


**Figura 4.31:** Documentos más relacionados al topic Belelú.

**Fuente:** Elaboración propia.

Junto al botón Inferir Labels se encuentra el botón *Mostrar Distribución Temática*, ver figura 4.29. Al hacer click en él se despliega el siguiente gráfico que muestra qué porcentaje tiene cada temática encontrada respecto a las demás, ver figura 4.32.

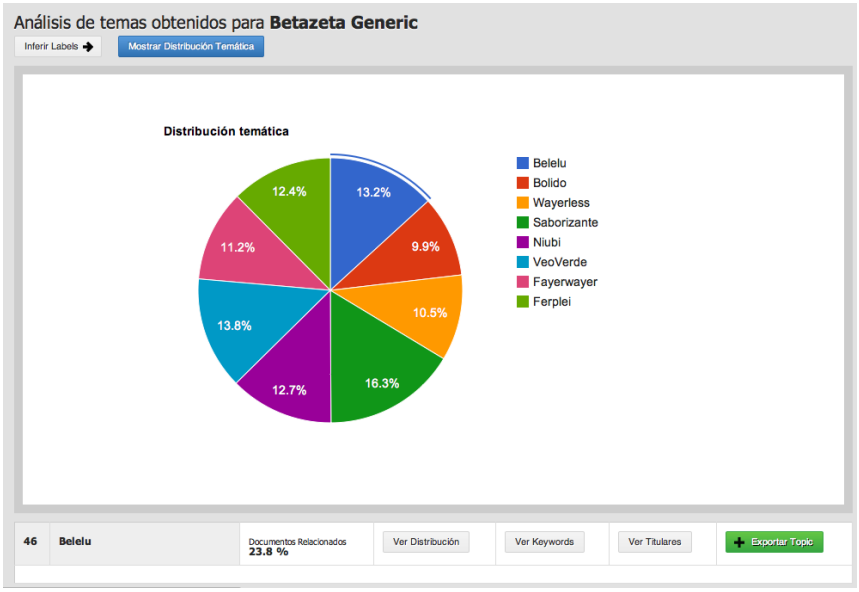


**Figura 4.32:** Distribución temática sobre Betazeta.

**Fuente:** Elaboración propia.

En general, para un modelo que entrena los 8 blogs se obtienen porcentajes similares. A continuación se muestran dos ejemplos, específicos a FayerWayer y Veoverde, que muestran la distribución temática de cada uno en las figuras 4.33 y 4.34 respectivamente.

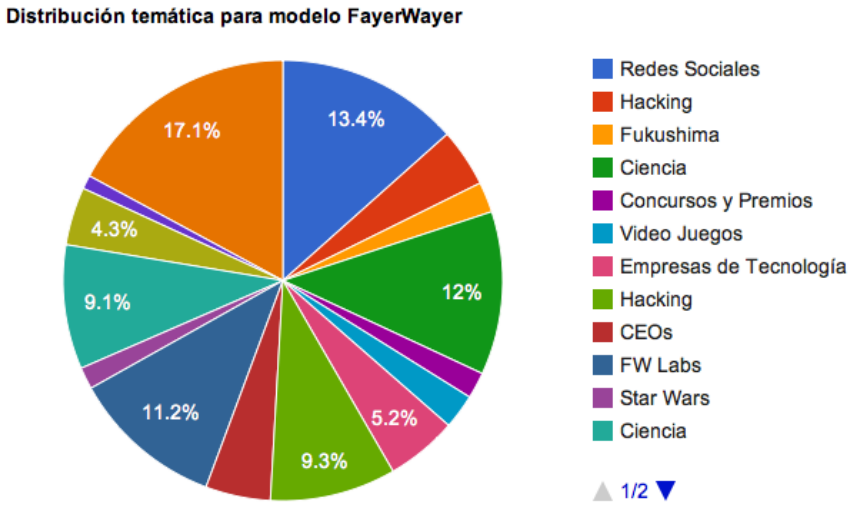


**Figura 4.33:** Distribución temática sobre FayerWayer.

**Fuente:** Elaboración propia.

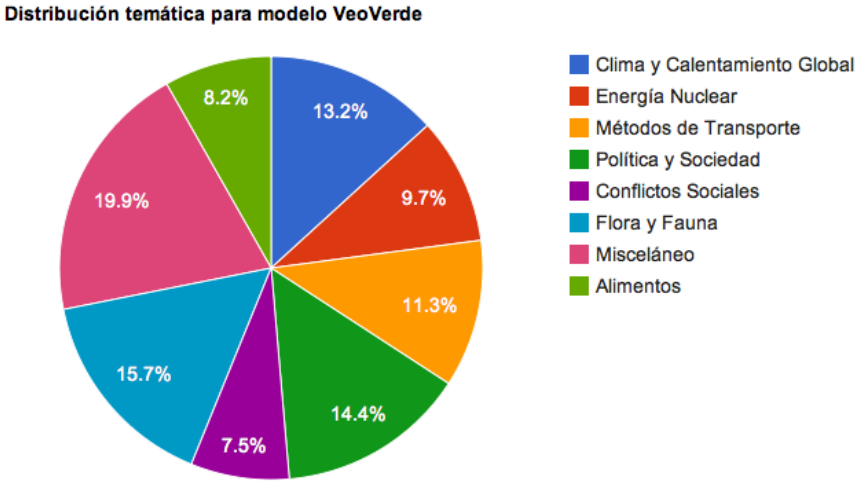


**Figura 4.34:** Distribución temática sobre VeoVerde.

**Fuente:** Elaboración propia.

Esta forma de visualizar los porcentajes de temáticas permite conocer cómo se distribuye el contenido que analiza el cliente. En el caso de Betazeta es difícil conocer a ciencia cierta qué contenido tienen, ya que es imposible leer manualmente cada artículo y generar un informe

en base a ello dado el costo en recursos que requiere. Si bien esta herramienta depende de la configuración del modelo y el criterio para asignar los nombres a las temáticas, entrega una base para poder entender el contenido. Para generar los gráficos de torta se utilizó Google Chart Tools[10].

Para comprender cómo funciona un modelo y el clasificador se provee de una pantalla que permite analizar un documento de la base de datos y ver cómo se distribuyen sus palabras respecto a cada temática. Para ello se debe hacer click en en el botón de ver documentos, que muestra los documentos más relacionados a un topic, ver figura 4.31, y luego presionar en el título del artículo, esto llevará al usuario a la pantalla de la figura 4.35.

En el detalle del post es posible ver el título del artículo, la clasificación que el modelo predictor le otorga dentro de los topics obtenidos, el texto original, el texto después de quitar Stopwords y otros caracteres y el texto luego de realizar stemming. Bajo lo anterior aparece el listado de topics y el porcentaje de asociación que tiene el texto con cada uno. Si se hace click en el nombre de uno de los topics se desplegará una nube de tags con las palabras del texto. El tamaño y color de cada palabra indica mayor o menor relación con un tema. Más claro y pequeño indica menor correlación con el topic. Cuando una palabra aparece en rojo indica asociación alta, ver figura 4.36.

[10] http://code.google.com/intl/es/apis/chart/


**Figura 4.35:** Detalle de un documento analizado.

**Fuente:** Elaboración propia.

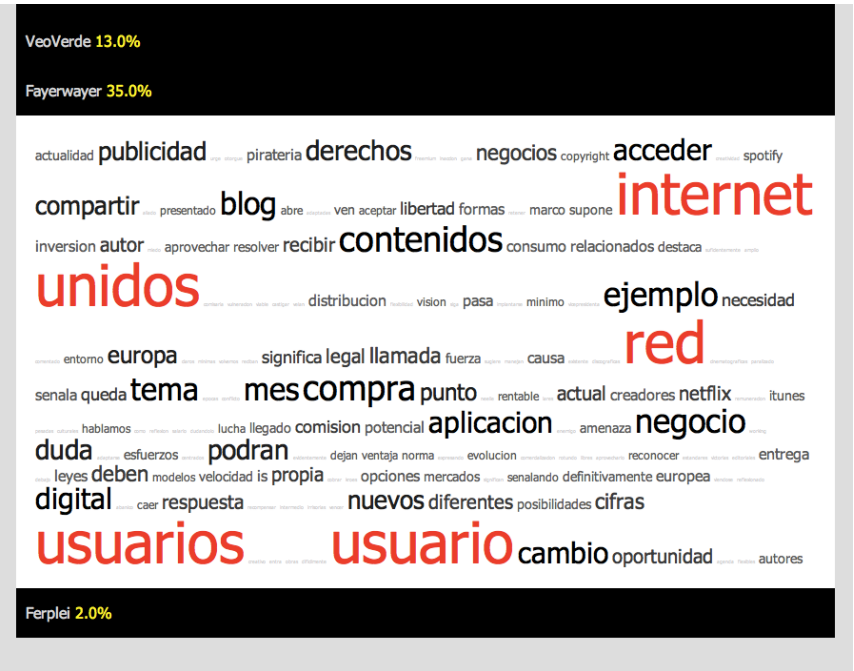


**Figura 4.36:** Nube de tags para el topic Fayerwayer.

**Fuente:** Elaboración propia.

### 4.4.4. Clasificador

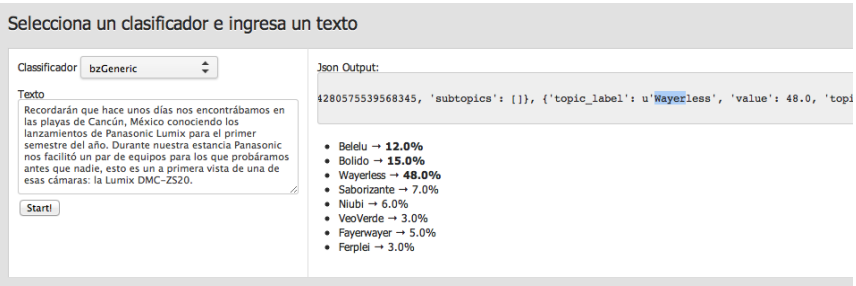


**Figura 4.37:** Sistema para pruebas de clasificación.

**Fuente:** Elaboración propia.

La función de este módulo es realizar pruebas sobre el comportamiento del clasificador. En él es posible ingresar un texto plano y seleccionar un clasificador. Internamente se hace una

consulta al webservice de clasificación enviando los parámetros seleccionados. El resultado es el *Json* que retorna el servicio y además la distribución porcentual entre las temáticas, ver figura 4.37.

El *Json* retornado por el webservice posee la siguiente estructura:

```
[
  {
   "topic_id":46,
   "topic_label":"TopicA",
   "value":12.0,
   "quality":1.1918465227817745,
   "keywords":[
     "nadie",
     "año",
     "igual",
     "realidad",
     "realmente"
   ],
   "subtopics":[
          {
           "topic_id":87,
           "topic_label":"SubTopicA",
           "value":27.0,
           "quality":1.1918465227817745,
           "keywords":[
             "casa",
             "año",
             "palabra",
             "realidad",
             "auto"
           ],
           "subtopics":[]
          },
          ...
   ]
  },
  ...
]
```

Donde *topic_id* es el id del topic en Tanalyzer, *topic_label* es el nombre asignado a dicha temática, *value* es el porcentaje de relevancia que posee un topic respecto a los otros que pertenecen al mismo nivel en el modelo analizado, *quality*[11] es el puntaje total del topic dividido en la cantidad de palabras analizadas. *keywords* son las n palabras más representativas de la temática asociadas al texto analizado, en el ejemplo n es cinco. *Subtopics* contendrá datos en el caso de que el clasificador posea un modelo asociado a ese topic de acuerdo al árbol de modelos antes descrito y posee exactamente los mismos parámetros que su predecesor.

[11] Más detalles sobre este valor se pueden encontrar en el Capítulo 5

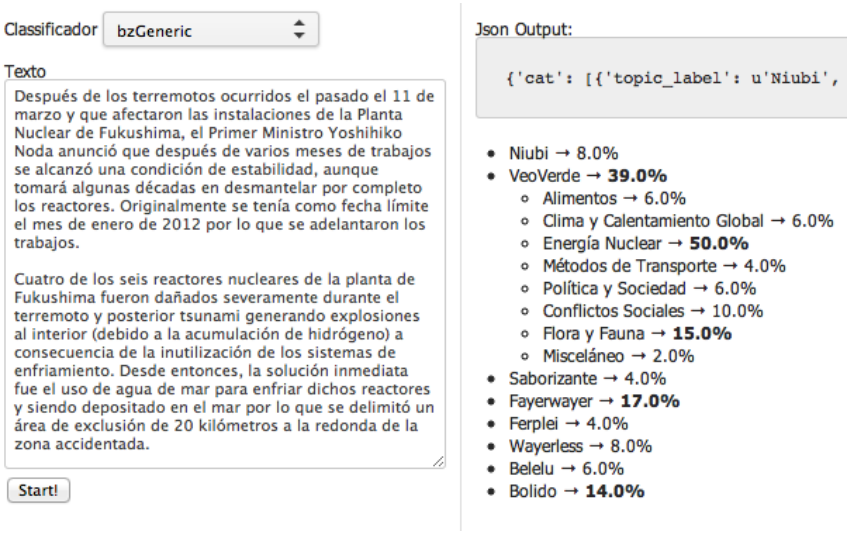


**Figura 4.38:** Clasificación usando múltiples modelos.

**Fuente:** Elaboración propia.

Cuando existen múltiples niveles en el clasificador el resultado se despliega como en la figura 4.38. En este caso se analiza un texto que habla sobre el desastre nuclear en Fukushima. El artículo es clasificado en VeoVerde y como subtopic se clasifica en Energía Nuclear con quality en 4.6, lo que indica que es una clasificación confiable.

### Consultas al WebService de manera remota

Para poder utilizar el Webservice de manera remota, asumiendo que la aplicación se encuentra corriendo en http://www.betazeta.com/, basta hacer un request via POST a la url:

```
http://www.betazeta.com/ws/betazeta/
```

Enviando los parámetros *classifier* y *text*, donde el primero representa el id del clasificador que se intenta utilizar y el segundo parámetro corresponde al texto que se evaluará. Un ejemplo de lo anterior se puede realizar utilizando el comando CURL disponible en sistemas Unix ejecutando en un terminal:

```
curl -X POST -d 'classifier=1&text=ipad' POST http://localhost:8000/ws/betazeta/
```

El resultado se puede observar en la figura 4.39.

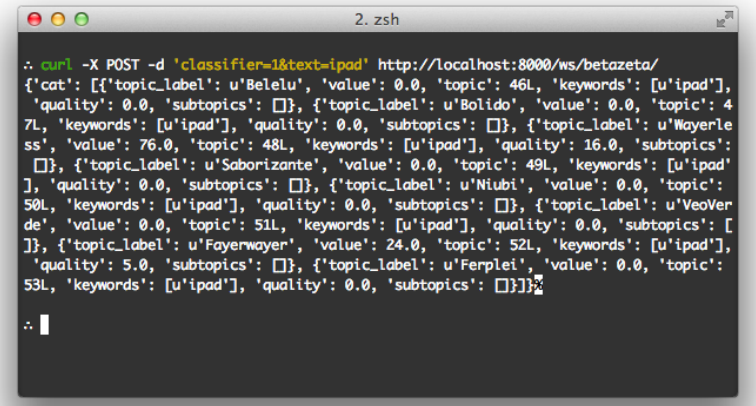


**Figura 4.39:** Consulta al webservice en forma remota.

**Fuente:** Elaboración propia.

### 4.4.5. Importar Datos

Éste módulo permite ejecutar el proceso Extract antes descrito. En la figura 4.40 se observa la interfaz para realizar scrapping.

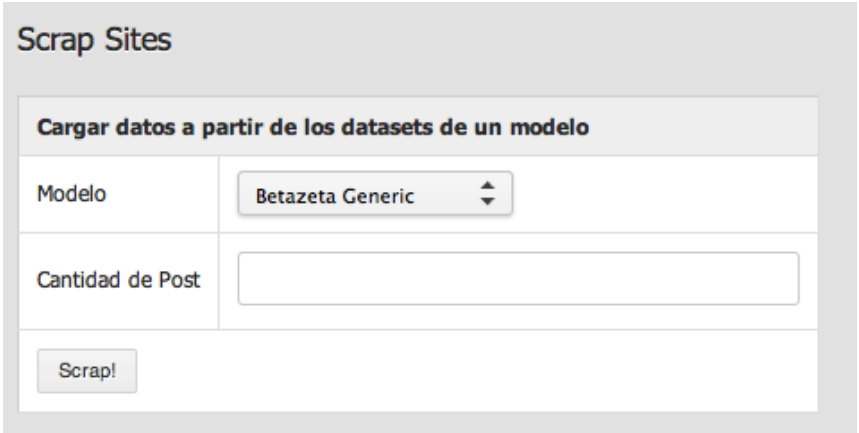


**Figura 4.40:** Modo para realizar scrapping de los sitios web de Betazeta.

**Fuente:** Elaboración propia.

El sistema de scrapping trabaja sobre un LdaModel. Se toman los DataSets asociados a dicho modelo y se consulta la url definida a esa entidad. Como parámetro se entrega el número de artículos a obtener en cada blog.

En la figura 4.41 se muestra la interfaz para realizar consultas directamente a la base de datos de Betazeta. En el lado izquierdo se definen los datos de conexión, entre ellos están el host, usuario, password, nombre de la base de datos y la tabla a obtener. En el lado derecho se definen los nombres de las columnas de la tabla que representan el título, contenido, id externo y url del artículo. Esto quiere decir que el texto ingresado en el campo título será el nombre de la columna que se copiará en el campo *title* del modelo Documents. Además se define a qué DataSet se asociarán los documentos obtenidos, y finalmente es posible realizar un filtro sobre la tabla objetivo para obtener sólo cierto tipo de datos. En el caso de Betazeta, que trabaja sus artículos sobre la plataforma Wordpress, es necesario obtener sólo los artículos publicados, pues existen muchos artículos de prueba o borradores de las versiones finales de los artículos, para ello se deben pedir sólo las publicaciones cuyo estado sea publicado, esto se puede realizar colocando *post_status = 'publish'* en el campo *Filtro Where*.

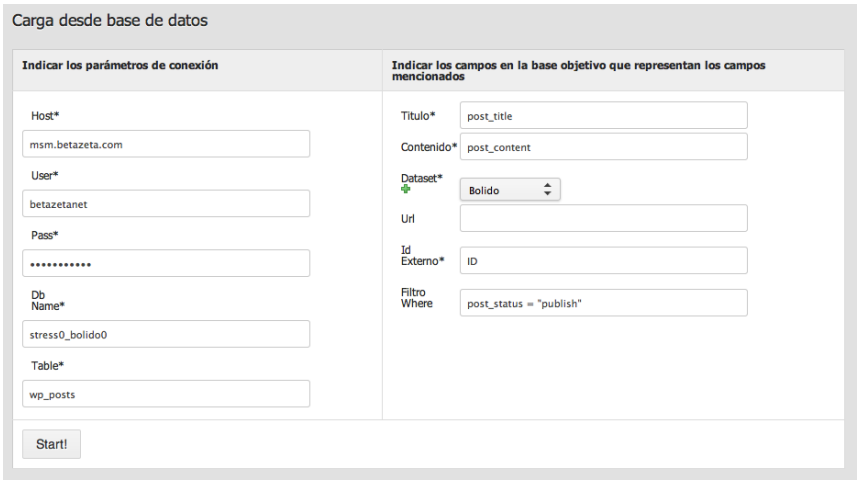


**Figura 4.41:** Interfaz para la obtención de artículos utilizando conexión directa a la base de datos.

**Fuente:** Elaboración propia.

### 4.4.6. Validación

Este módulo fue incorporado exclusivamente para validar los modelos, pues no constituye un requisito formal de la empresa, pero permite simplificar la tarea de clasificar publicaciones en forma manual para luego contrastar estos resultados con las predicciones de un modelo y entender en qué configuraciones permiten realizar predicciones de confianza y bajo qué condiciones son aceptables o no.

Validación consta de dos etapas, en la primera se selecciona el modelo a evaluar, ver figura 4.42, y en la segunda etapa se muestra al usuario un texto al azar, elegido dentro del conjunto de validación, el cual se podrá asociar a uno o más topics del modelo seleccionado, ver figura 4.43. Una vez evaluado, el

texto no aparecerá dentro de la selección aleatoria nuevamente. Esto se realiza hasta que ya no quedan más artículos por categorizar. Finalizado este proceso se ejecutan los métodos de cáluclo de *Precision*, *Recall* y *F-Measure*, lo que se profundiza en el capítulo que se presenta a continuación.

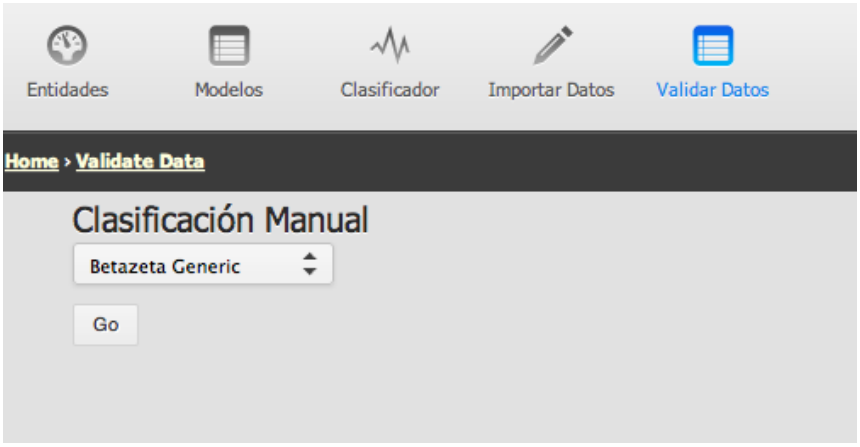


**Figura 4.42:** Selección del modelo a validar.

**Fuente:** Elaboración propia.


**Figura 4.43:** Interfaz para la clasificación manual de documentos.

**Fuente:** Elaboración propia.

# Capítulo 5

# Experimentos

En este capítulo se presentan los diferentes experimentos realizados. En primer lugar se describe el conjunto de datos, se detalla el cálculo de las medidas de efectividad respecto a los datos y luego se detallan las diferentes configuraciones adoptadas para efectuar las pruebas y su análisis respectivo en cada una de ellas.

## 5.1. El conjunto de datos

Para confeccionar un modelo de pruebas suficientemente robusto se descargan 4.000 documentos por cada uno de los ocho blogs en estudio, generando un total de aproximadamente 32.000 textos diferentes. Estos datos se obtienen el día 20 de enero de 2012 utilizando el método de scrapping antes mencionado. Se destaca que no hay una selección especial en el conjunto de artículos obtenidos por cada blog, éstos simplemente se ordenan por fecha de aparición.

Dada la frecuencia de publicación de cada blog, el periodo comprendido por estos 4.000 artículos varía. Ferplei es el blog con mayor cantidad de publicaciones mensuales, en este caso los documentos obtenidos abarcan sólo cuatro meses mientras que Veoverde o Saborizante alcanzan a cubrir 15 y 16 meses respectivamente. Para este análisis se consideraron los meses de octubre, noviembre y diciembre de 2011 pues se contaba con todos los post publicados en esas fechas para los ocho blogs y con ello se obtuvo un promedio de publicaciones mensuales con los que se realizó la proyección, ver tabla 5.1.

| | Ferplei | Wayerless | Fayer Wayer | Bolido | Niubi | Belelu | Veoverde | Saborizante | Todos |
|---|---|---|---|---|---|---|---|---|---|
| Diciembre 2011 | 945 | 425 | 451 | 309 | 333 | 301 | 273 | 253 | 3290 |
| Noviembre 2011 | 881 | 439 | 442 | 349 | 326 | 278 | 273 | 252 | 3240 |
| Octubre 2011 | 848 | 417 | 325 | 342 | 359 | 237 | 246 | 223 | 2997 |
| Promedio | **891** | **427** | **406** | **333** | **339** | **272** | **264** | **243** | **3275** |
| Meses Obtenidos | 4 | 9 | 10 | 12 | 12 | 15 | 15 | 16 | - |

**Cuadro 5.1:** Artículos publicados por blog durante los meses de Octubre, Noviembre y Diciembre de 2011 y número de meses de datos obtenidos con 4000 documentos por blog en base a las tasas de publicación.

**Fuente:** Elaboración propia.

Luego de analizar los datos en forma manual se observan múltiples documentos vacíos. Por ejemplo, en el caso de Saborizante esto ocurre debido a que el contenido del artículo consiste en imágenes que representan un afiche de un evento, la cual se ignora en el proceso de extracción de texto. Lo mismo ocurre en Fayerwayer cuando se publica el cómic 'Juanelo'. En general publicaciones con textos muy cortos cuyo foco son imágenes o videos son eliminados obteniendo la cantidad de documentos por blog que se muestra en la tabla 5.2.

| | Fayerwayer | Wayerless | Ferplei | Belelu | Niubi | VeoVerde | Bolido | Saborizante |
|---|---|---|---|---|---|---|---|---|
| Publicaciones | 3990 | 3929 | 3940 | 3946 | 4000 | 4000 | 3940 | 3842 |

**Cuadro 5.2:** Número de artículos analizables después del proceso de limpieza de documentos.

**Fuente:** Elaboración propia.

## 5.2. Cálculo de medidas de efectividad

Como se menciona anteriormente las medidas de efectividad son *Precision*, *Recall* y *F-Measure*. En particular se utiliza *F-Measure* sin variaciones, esto significa que ni *Precision* o *Recall* son ponderados. Para calcular estas medidas se define un proceso de validación dividido por etapas.

### 5.2.1. Clasificación de post

Se recuerda que antes de realizar el proceso de entrenamiento se separa un porcentaje de los artículos que no se utilizan para generar el modelo y se almacenan para hacer pruebas posteriores. En esta etapa se toman estos textos separados para validación y se les asigna una clasificación de acuerdo al modelo entrenado.

La predicción sobre cada documento contendrá datos para todos los topics del modelo. Cada predicción indica el porcentaje de correlación del texto con cada tópico y su quality, indicador que se describe a continuación.

Para determinar que una clasificación es correcta, es necesario definir un margen de tolerancia sobre dos parámetros. El primer parámetro es el porcentaje de correlación. Si el porcentaje de correlación está bajo un valor, entonces se entiende que la clasificación se rechaza y por sobre éste se acepta. El segundo parámetro es Quality o la calidad de la predicción, la idea detrás de este parámetro es poder tener un indicador que permita eliminar algunos casos de falsos positivos. En un modelo con dos temáticas es posible tener dos textos que posean distribuciones porcentuales idénticas pero con distinto significado. Por ejemplo, ambos textos pueden tener una correlación del 80 % con el Tema A y 20 % con el Tema B, pero los valores del clasificador pueden ser de 8000 y 2000 para el primer texto y en el segundo 8 y 2, indicando que el segundo texto en realidad no está asociado a ninguno de los dos temas. Quality busca desambiguar estos casos utilizando el valor real entregado por el clasificador. Se destaca que para un texto habrá un valor de Quality por cada Topic.

### 5.2.2. Clasificación de referencia

Además de la clasificación realizada por Tanalyzer es necesario contar con una clasificación de referencia, no necesariamente balanceada, la que se asume correcta para contrastar con los resultados obtenidos por el algoritmo y de ésta forma calcular los indicadores mencionados anteriormente. En este caso se almacena el modelo, el documento y el o los topics asociados al documento.

### 5.2.3. Cálculo de Precision

Como se menciona, el algoritmo determina que una predicción es correcta cuando su porcentaje de asociación dentro de los topics tiene un valor superior a una tolerancia y una calidad (quality) aceptable. De acuerdo a la fórmula 3.9, las predicciones positivas correctas se definen como el grupo de predicciones positivas - sobre el umbral - realizadas por Tanalyzer y que coinciden con lo indicado por la clasificación de referencia. Las clasificaciones positivas son todas las clasificaciones que hace el software y que están por sobre el umbral de tolerancia.

Para obtener *Precision* sobre un modelo se calcula el indicador sobre cada clase del modelo y luego se promedian sus valores, además es importante destacar que para los cálculos siempre se toman sólo los documentos que fueron evaluados en el conjunto de referencia, es decir, si un documento fue separado para test, posee una clasificación hecha por el algoritmo, pero no posee una clasificación de referencia, entonces no se inlcuye en los cálculos de ningún indicador.

### 5.2.4. Cálculo de Recall

De la misma manera, según la fórmula 3.10, las predicciones positivas correctas se trabajan exactamente como en *Precision*. Los documentos de clase positiva, por otro lado, corresponden a las categorías reales de los documentos, es decir, el conjunto de clasificación de referencia completa.

Al igual que en *Precision*, *Recall* obtiene un valor para cada topic y luego estos se promedian para obtener el *Recall* del modelo completo, asímismo, el universo de documentos analizado siempre está acotado por el set de clasificación de referencia.

## 5.3. Experimentos y análisis de resultados

A continuación se describen los distintos tipos de análisis efectuados al software.

### 5.3.1. Clasificación de referencia automática

Sobre los siguientes experimentos se realiza clasificación automática. Para ello se preparan distintos modelos que permiten obtener exactamente un topic por cada sitio o DataSet analizado, gracias a la alta diferencia que los contenidos de los Blogs presentan entre sí. De ésta forma es posible establecer una relación DataSet-Topic 1 a 1, por ejemplo, al analizar 8 DataSet los modelos se configuran de manera que se obtenga un topic para cada sitio y así sea posible relacionar el topic 'Tecnología' a 'FayerWayer', 'Medio Ambiente' a 'Veoverde' y así sucesivamente. Como cada documento está asociado a un DataSet inmediatamente es posible clasificarlo y asignarle un topic, quedando todos los post de un mismo DataSet en una categoría, por ejemplo, todo Belelú quedaría clasificado en 'Mujer'.

#### Análisis de sensibilidad sobre Alpha y Beta

El análisis de sensibilidad consiste en probar diferentes valores para los parámetros ajustables en un modelo y comparar cómo afectan los resultados. Los valores seleccionados para Alpha son 0.09, 0.1, 0.3, 0.4 y 0.5 con Beta fijo en 1 y para Beta se utilizaron los valores 2, 1, 0.9, 0.7, 0.5 y 0.4 con Alpha fijo en 0.1. Valores mayores o menores a los antes mencionados no generan la relación 1:1 Blog-Topico, pues aparecen subtópicos de algunas áreas en lugar de un macro-tópico y por lo tanto no se utilizaron para este análisis. Los resultados se muestran a continuación.

Como se menciona en secciones anteriores, un texto se considera relacionado a un tema cuando posee un alto porcentaje de asociación sobre éste. El criterio de selección consiste en fijar un punto de corte. Si el texto posee un porcentaje de asociación mayor o igual al punto de corte se considera dentro de la clase.

Las figuras 5.1, 5.2, 5.3 muestran cómo cambian las medidas de *Precision*, *Recall* y *F-Measure* variando alpha respecto a diferentes valores del punto de corte (eje $x$). Como es de esperar, cuando el porcentaje de corte es muy bajo, se considerará que el texto pertenece a muchos topics, por consiguiente se tendrán muchos falsos positivos, esto implica que *Precision* será bajo. Por otro lado, al incluir el texto en muchas clases es muy probable que entre ellas se encuentre la clasificación correcta, por este motivo *Recall* será alto. Lo opuesto ocurre al aumentar el porcentaje de corte, como se observa en los tres gráficos.

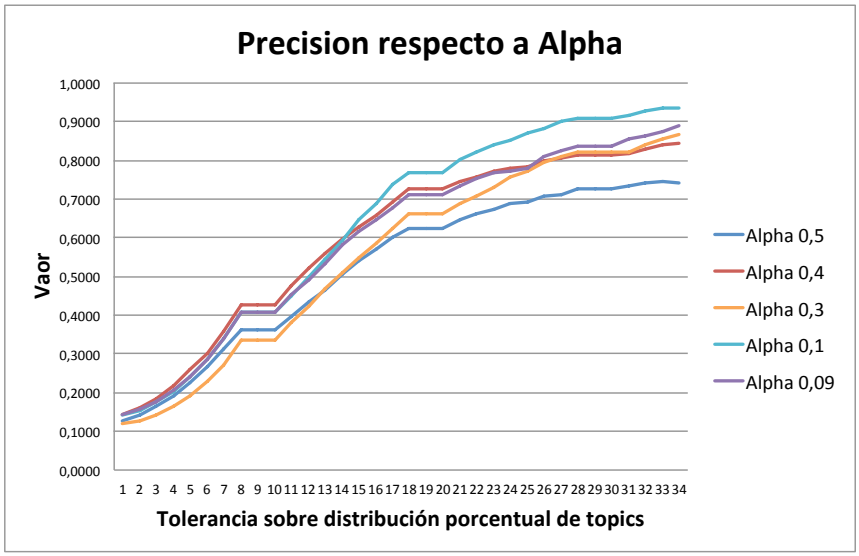


**Figura 5.1:** Precision respecto a distintos valores de alpha.

**Fuente:** Elaboración propia.

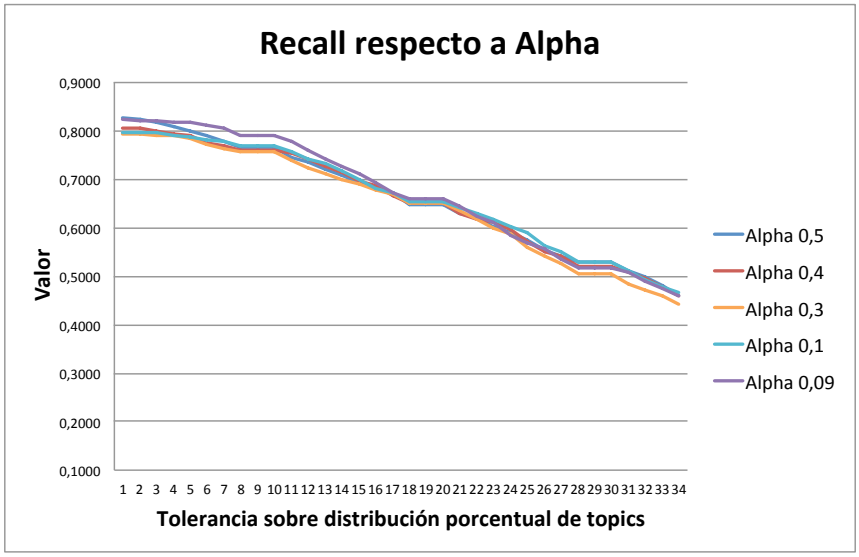


**Figura 5.2:** Recall respecto a distintos valores de alpha.

**Fuente:** Elaboración propia.

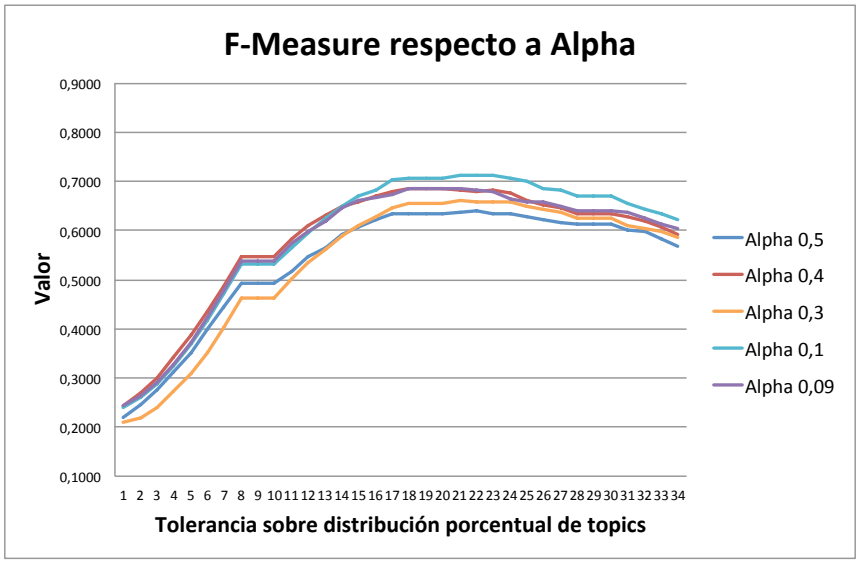


**Figura 5.3:** F-measure respecto a distintos valores de alpha.

**Fuente:** Elaboración propia.

De la misma manera, para Beta se obtienen los resultados visibles en las figuras 5.4, 5.5, 5.6.

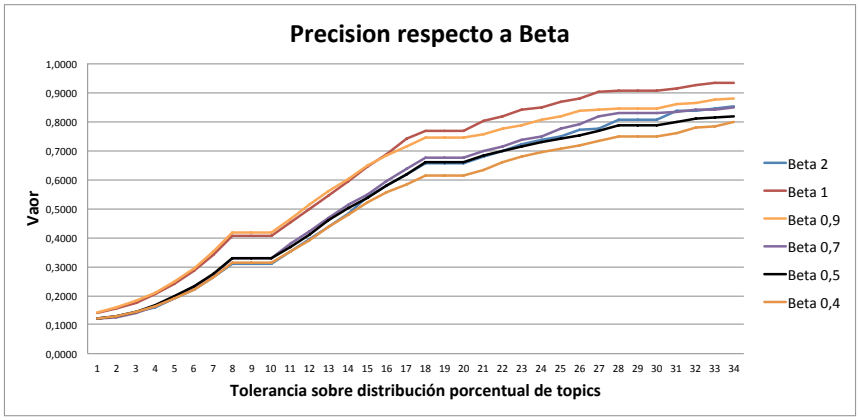


**Figura 5.4:** Precision respecto a distintos valores de beta.

**Fuente:** Elaboración propia.

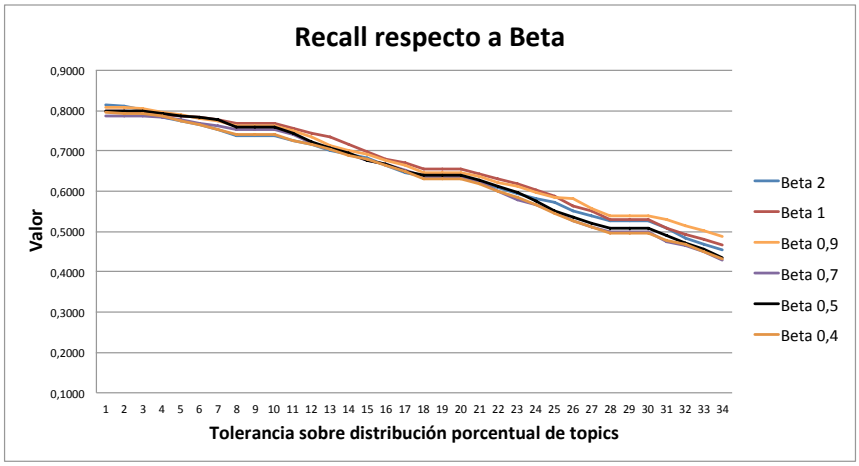


**Figura 5.5:** Recall respecto a distintos valores de beta.

**Fuente:** Elaboración propia.

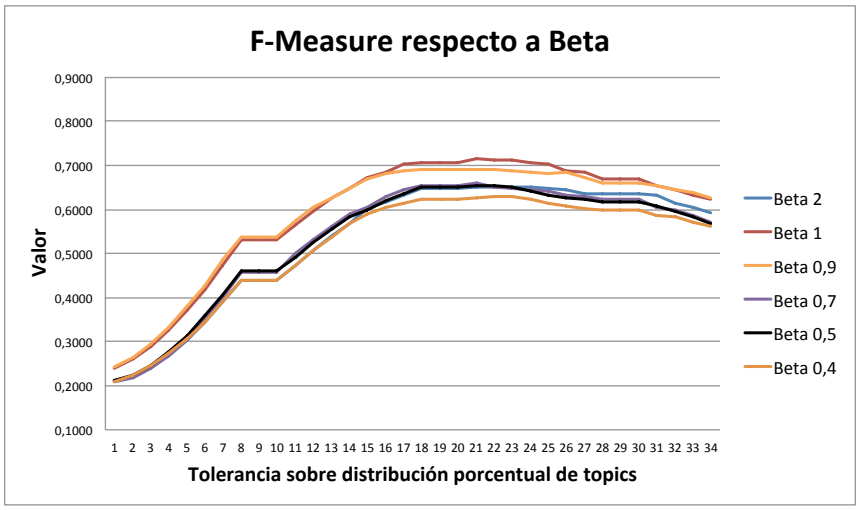


**Figura 5.6:** F-measure respecto a distintos valores de beta.

**Fuente:** Elaboración propia.

Como se puede observar en los gráficos 5.3 y 5.6, para un punto de corte de 21 % usando Alpha 0.1 y Beta 1 se obtiene un valor máximo de *F-Measure* de 71,3 %. Para ver en detalle las tablas de datos que generan cada gráfico consultar el Anexo.

### Análisis de efectividad filtrando por quality

En este caso el filtro por Quality se utiliza distinto al filtro por porcentaje de asociación. Cuando un documento posee bajo Quality en todos los topics implica normalmente que no es posible asegurar un resultado confiable. Para un texto se tienen múltiples valores de Quality, uno por Topic. Se selecciona el mayor. Si este valor es menor al punto de corte para Quality entonces el documento se descarta del análisis, pues no es posible asegurar nada respecto a su contenido. Estos casos deben ser revisados y clasificados en forma manual.

En la figura 5.7 es posible observar la evolución de *Precision*, *Recall* y *F-Measure* para distintos valores de quality utilizando un modelo entrenado con Alpha 0.1, Beta 1 con un punto de corte de porcentaje de asociación definido en 21 %.

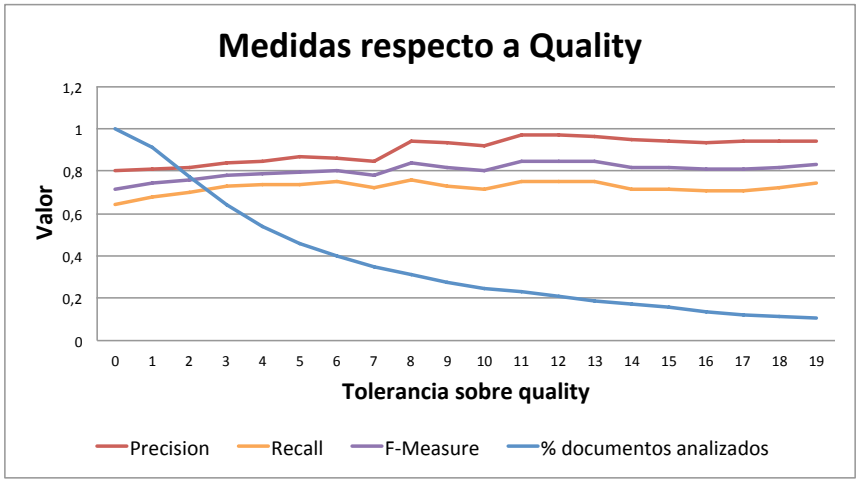


**Figura 5.7:** Evolución de F-measure, Precision y Recall respecto a filtrar documentos por Quality y el porcentaje de documentos analizados respecto al total original.

**Fuente:** Elaboración propia.

En el gráfico se observa una tendencia a obtener una clasificación más efectiva al analizar documentos que posean un mayor nivel de Quality, el problema es que este filtro es sumamente estricto disminuyendo considerablemente la cantidad de documentos que son analizados. Para este modelo parece aceptable definir un nivel de corte de 2 puntos de Quality, mejorando *F-Measure* de un 71,3 % a 75,5 % permitiendo clasificar el 78 % de los documentos en forma automática. Se destaca además que si un documento posee nivel de Quality alto es posible asegurar una clasificación confiable, alcanzando incluso 96 % de *Precision* con 84 % de *F-Measure*.

### Ánalisis de efectividad aumentando el número de iteraciones

En la descripción de la librería de LDA utilizada se menciona la opción de definir el número de iteraciones que realiza el algoritmo. Por defecto se utilizan 50 iteraciones en forma constante para

cualquier entrenamiento. A continuación se compara *Precision*, *Recall* y *F-Measure* con Alpha 0.1 y Beta 1 para 50 y 500 iteraciones. Los resultados se pueden observar en las figuras 5.8, 5.9 y 5.10.

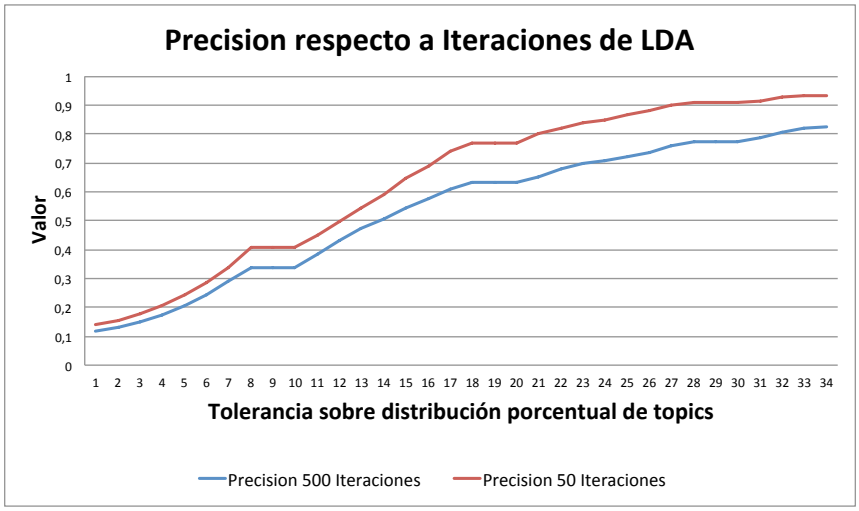


**Figura 5.8:** Evolución de Precision respecto a 50 y 500 iteraciones.

**Fuente:** Elaboración propia.

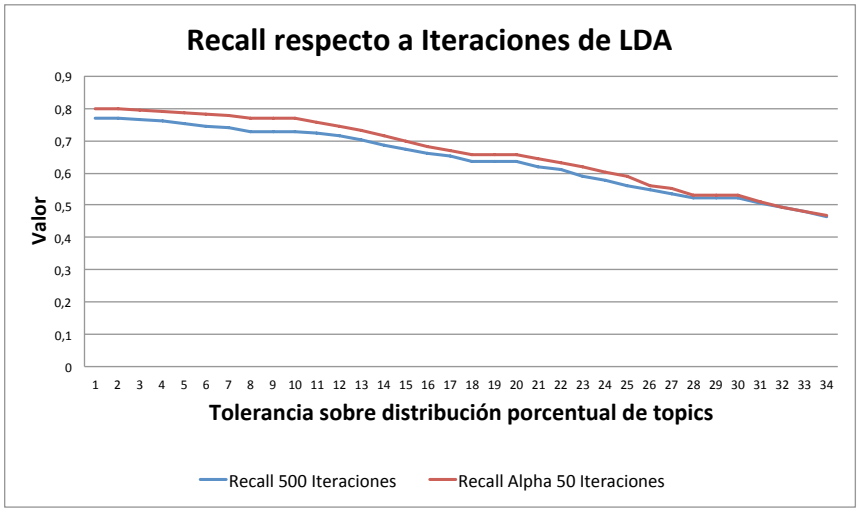


**Figura 5.9:** Evolución de Recall respecto a 50 y 500 iteraciones.

**Fuente:** Elaboración propia.

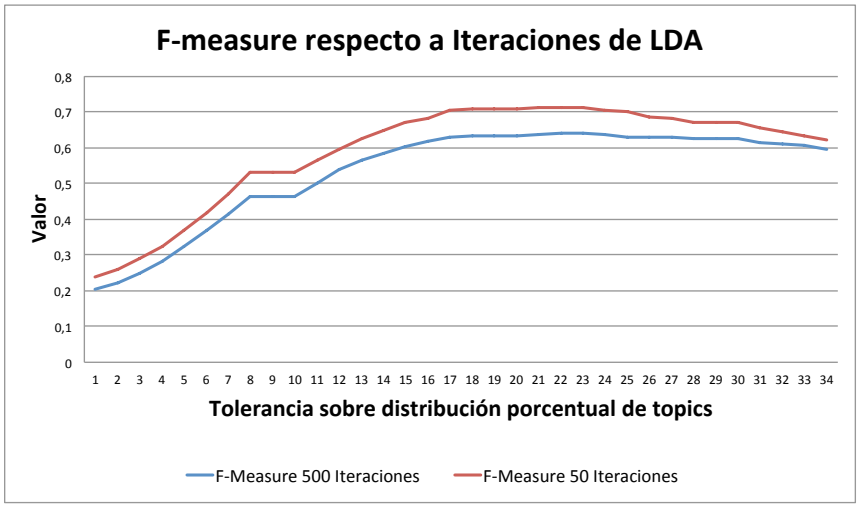


**Figura 5.10:** Evolución de F-measure respecto a 50 y 500 iteraciones.

**Fuente:** Elaboración propia.

A pesar de lo que se podría esperar, aumentar el número de iteraciones de LDA disminuye la efectividad del modelo. Como se observa en los gráficos tanto *Recall* como *Precision* disminuyen su efectividad, siendo éste último el más afectado. La disminución de *Recall* se puede deber a un aumento en la cantidad de falsos negativos, y la disminución de *Precision* a un aumento en los falsos positivos.

Ambos factores se pueden explicar en base a que cuando se hacen más iteraciones el algoritmo tiende a especializarse, obteniendo clases que son menos genéricas. Un ejemplo extremo del fenómeno anterior podría ser que el tema *Tecnología* en realidad está representando sólo los temas de tecnología asociados a *Procesadores* en lugar de la gama completa de Tecnología. Siguiendo el ejemplo, esto causa que un post de Tecnología que no habla de Procesadores sería descartado del tema, aumentando los falsos negativos. Si además posee un porcentaje de correlación mayor con otro tema será clasificado erróneamente aumentando los falsos positivos.

Es importante destacar que el uso de más iteraciones para generar mayor especificidad puede ser útil para la generación de buenos submodelos dentro de un área. El problema de este método es que requiere realizar más operaciones aumentando su costo computacional. En este caso partícular el tiempo de ejecución aumento cuatro veces.

### Análisis sobre blogs con alta y baja correlación

Es interesante estudiar cómo se comporta el clasificador cuando se entrena usando DataSets con alta y baja correlación. Para ello se toman dos pares de DataSets, el primero contiene sólo los artículos de FayerWayer y Wayerless, dos blogs altamente relacionados. El segundo contiene los artículos de Niubie y Ferplei, blogs con baja correlación. Es esperable que en el caso del primer modelo sea más difícil para el clasificador distinguir entre uno y otro sitio, mientras que lo opuesto debería ocurrir en el caso del segundo modelo. Los resultados se pueden ver en las figuras 5.11, 5.12 y 5.13.

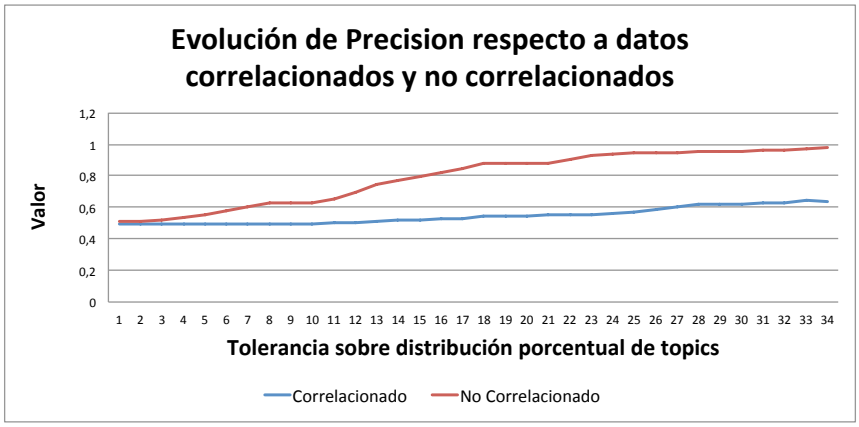


**Figura 5.11:** Comparativa de Precision sobre set de datos correlacionados y no correlacionados.

**Fuente:** Elaboración propia.

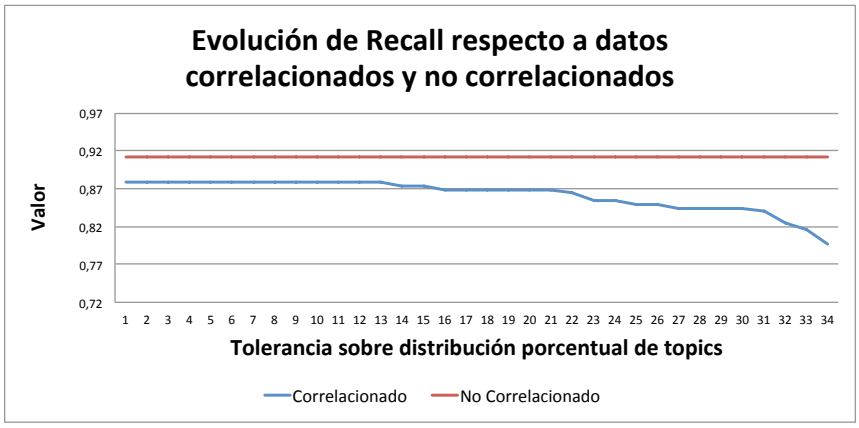


**Figura 5.12:** Comparativa de Recall sobre set de datos correlacionados y no correlacionados.

**Fuente:** Elaboración propia.

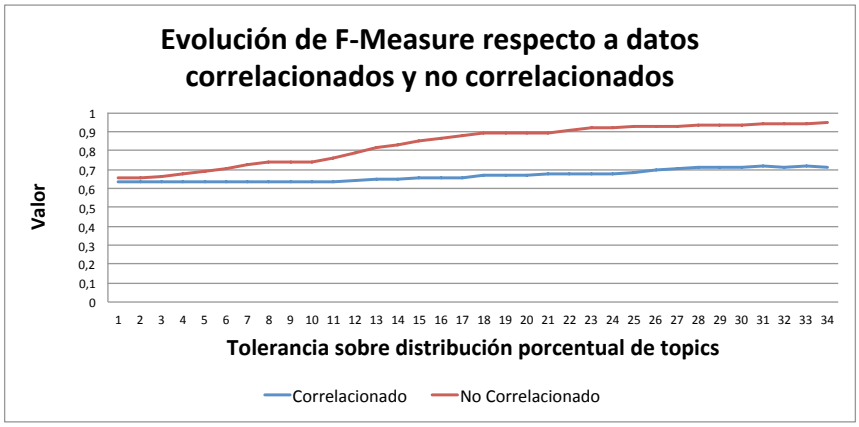


**Figura 5.13:** Comparativa de F-Measure sobre set de datos correlacionados y no correlacionados.

**Fuente:** Elaboración propia.

Como es de esperar, el modelo no correlacionado obtiene excelentes resultados. Se destaca que *Recall* y *Precision* en ambos modelos poseen altos valores, mayores que en otros experimentos que obtienen 8 Topics en lugar de 2. La explicación es muy simple, esto se debe a que el número de Falsos Negativos y Positivos es menor, ya que sólo hay dos clases de donde elegir la clasificación correcta. En el caso de modelos no correlacionados se tiene además que los porcentajes de distribución generalmente van a ser menos homogéneos y alcanzarán altos valores, facilitando la diferenciación entre resultados positivos y los que no. Utilizando el segundo modelo se clasifica el siguiente texto obtenido de un artículo de Niubie:

> *Hace unos días comentábamos que Blizzard temía no cumplir las expectativas con Diablo III, gracias a un mensaje "sarcástico" del Gerente de Comunidades. Pues resulta que nadie entendió la "broma" y el tema se salió de control, de menos eso pensamos al ver la nueva carta que escribió Jay Wilson a la comunidad.*
>
> *Para aquellos que se preguntan cuándo saldrá a la venta, Jay promete que pronto ofrecerán detalles de la fecha de lanzamiento. De igual modo dice que si este capítulo no fuera un digno sucesor de la saga Diablo ni siquiera lo liberarían.*

Se obtienen los siguientes porcentajes de distribución: Video Juegos: 74 %, Fútbol: 26 %. En este caso, hay que definir un punto de corte muy alto, dado que de lo contrario ambas clases serán seleccionadas como categorías correctas, haciendo que *Recall* se mantenga constante para muchos valores del punto de corte, ya que siempre estarán incluidos los verdaderos positivos y no se ve afectado por falsos positivos, a diferencia de *Precision*.

Uno de los motivos por el cual *Recall* en el caso del modelo no correlacionado es mayor al correlacionado se debe a que en este modelo es mucho más difícil generar falsos positivos, pues normalmente la clase positiva tendrá valores de asociación al texto de entre un 60 % a 100 %, haciendo que el punto de corte tenga que ser sumamente alto para disminuir el *Recall* respecto a otros modelos.

| Topic obtenidos para FayerWayer | |
|---|---|
| **Label** | **Keywords** |
| Redes Sociales | facebook red sociales redes internet twitter social usuarios web espana |
| Falla en Servicios de Empresas | sony seguridad datos compania usuarios empresa problema hackers ataque ataques |
| Global | nuclear energia bin japon planta laden imagenes fukushima photoshop osama |
| Ciencia | años investigadores universidad cientificos investigacion estudio robot tecnologia energia desarrollo |
| Concursos y Premios | blusens kindle ganadores premio concurso libros amazon oro fayerwayer tv |
| Video Juegos | kinect proyecto videojuegos universidad estudiantes evento wikipedia desarrollo robot competencia |
| Empresas de Tecnología | internet empresas pais argentina empresa productos apple patentes paises servicios |
| Política y Sociedad | ley internet gobierno anonymous web sitio sitios derechos pais unidos |
| CEOs y Empresas | apple jobs steve compania empresa us años hp ceo año |
| Gadgets | pantalla 3d ces intel tecnologia pulgadas us samsung hd usb |
| Star Wars y Compras Online | comprar star us groupon wars producto precio descuento ebay cupones |
| Espacio | nasa tierra espacial espacio años planeta metros mision año agua |
| Música y Películas | apple musica itunes canciones peliculas store music tienda videos ipod |
| Linux | metro gnome kernel linux ubuntu linus madrid torvalds mint unity |
| Internet | google usuarios servicio microsoft compañia windows facebook web internet año |

**Cuadro 5.3:** Tabla que muestra las keywords y el label asociado para cada tópic en el modelo.

**Fuente:** Elaboración propia.

*Precision* también toma altos valores, dado que el número de Falsos Positivos a lo más es uno en ambos modelos a diferencia de cuando se trabaja con 8 clases, donde se puede llegar a tener 7 Falsos Positivos bajo las condiciones del experimento anterior. De aquí se desprende que a mayor número de clases en el mismo modelo, mayor es la dificultad de predecir correctamente.

### 5.3.2. Clasificación de referencia manual

Al analizar un área específica no es posible inferir un topic a partir del DataSet, pues todos los documentos poseen el mismo. En este caso se utiliza la interfaz para clasificación manual de topics descrita en el capítulo anterior.

#### Pruebas para un modelo específico

Para evaluar el comportamiento de la clasificación manual sobre un modelo específico se utiliza el blog FayerWayer, el más relevante dentro de la red de Betazeta. Se crea un modelo con Alpha 0.1 y Beta 1 y se configura para obtener 15 temáticas. Luego de entrenar el modelo se asignan labels en base a los keywords y titulares más relevantes asociados a cada topic, los que se pueden ver en la tabla 5.3.

Utilizando la interfaz de validación se clasifican 120 artículos de prueba en forma manual. Se asigna a cada artículo una o más categorías dentro de las 15 obtenidas por el modelo. Los resultados para *Precision*, *Recall* y *F-Measure* sin filtrar por Quality se pueden observar en la figura 5.14.

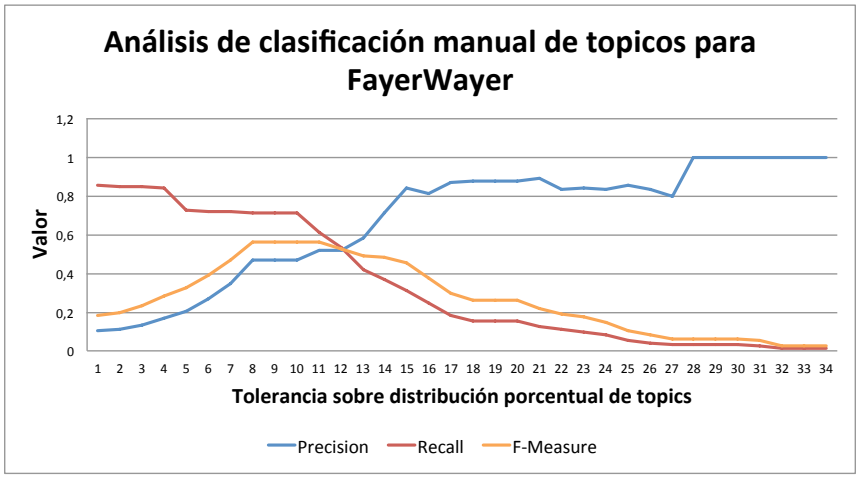


**Figura 5.14:** Comparativa de Precision, Recall y F-Measure sobre modelo entrenado utilizando sólo los artículos de FayerWayer.

**Fuente:** Elaboración propia.

Como se puede ver en los gráficos, los resultados son inferiores al modelo antes analizado con 8 blogs. Esto se explica debido a que este tipo de clasificación tiene un grado de complejidad mucho mayor. En primer lugar, los topics a evaluar no son fijos, la cantidad correcta de temáticas debe ser estimada y no necesariamente existe una correlación directa entre las temáticas que maneja un blog, es decir, las categorías en las que los autores clasifican manualmente los artículos y las obtenidos por un modelo, no necesariamente coinciden.

También se tiene que bajo este sistema un artículo puede poseer múltiples categorías y no sólo una, como en el modelo más general recién evaluado. Además de lo anterior, una vez que se obtiene el set de topics es fundamental definir muy bien sus nombres, dado que en base a esto se realizará la clasificación de referencia y si el nombre del topic no está bien definido puede provocar que la clasificación manual, que se asume correcta, no esté bien hecha.

Finalmente, un artículo no necesariamente va a pertenecer siempre a una de las categorías obtenidas por el modelo, ya que puede estar asociado a un micro tema que el modelo no fue capaz de detectar, esto provoca que al realizar la clasificación de referencia en forma manual no sea posible clasificar el artículo en las categorías obtenidas por el modelo, de ésta manera el clasificador intentará asignar una clasificación al artículo generando siempre un falso positivo.

En particular *Recall* presenta un valor muy bajo. Esto se explica debido al número de topics. Al tener 15 categorías con más de una clasificación correcta por cada artículo, cuando aumenta el punto de corte del porcentaje de participación, más resultados correctos quedan fuera de las clases seleccionadas, aumentando rápidamente el número de falsos negativos y por consiguiente disminuyendo el valor de *Recall*.

Al trabajar con más categorías los porcentajes de distribución van a ser más pequeños en general, aún para las clases correctas, haciendo más difícil distinguir entre una buena categoría y una que no lo

es. Por otra parte *Precision* se mantiene alto, como en los modelos anteriores, incluso para un punto de corte de 29 % ya es posible obtener un 100 % de precisión. El clasificador entonces, si bien deja clases correctas fuera del resultado, las que selecciona usualmente son verdaderos positivos.

Ahora bien, agregando el filtro por Quality al modelo se obtienen los gráficos 5.15,5.16 y 5.17 para *Precision*, *Recall* y *F-Measure* respectivamente.

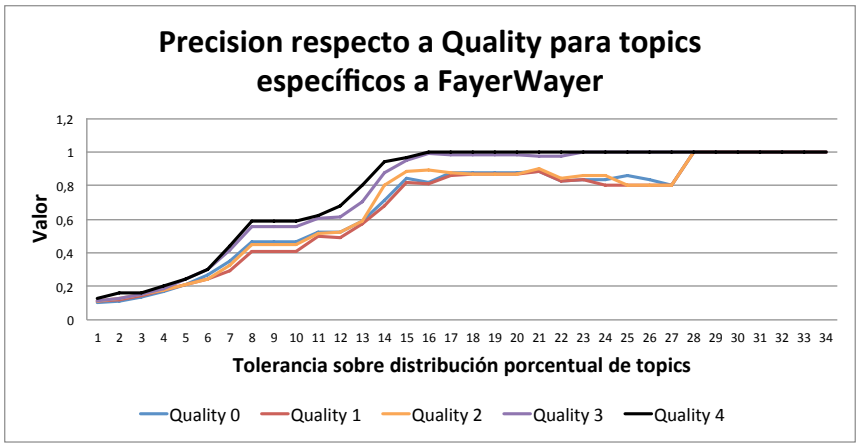


**Figura 5.15:** Análisis de Precision respecto a Quality para modelo entrenado utilizando sólo los artículos de FayerWayer.

**Fuente:** Elaboración propia.

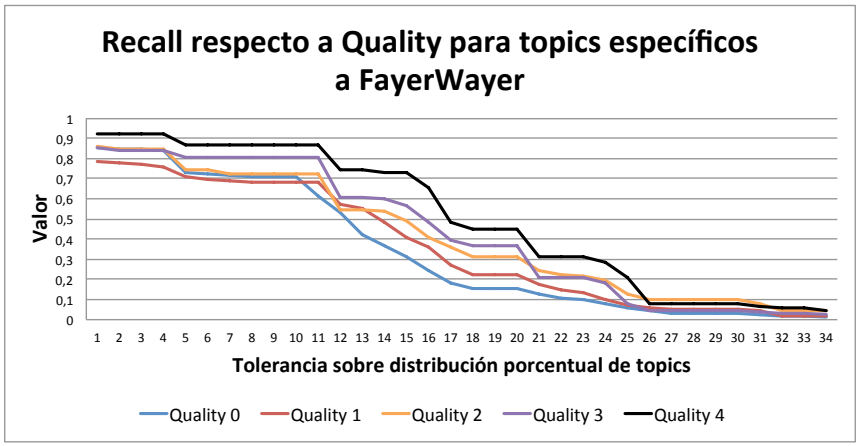


**Figura 5.16:** Análisis de Recall respecto a Quality para modelo entrenado utilizando sólo los artículos de FayerWayer.

**Fuente:** Elaboración propia.

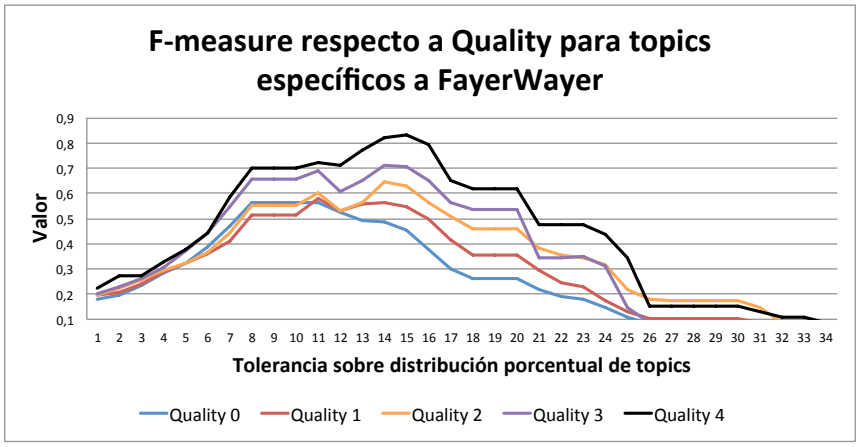


**Figura 5.17:** Análisis de F-Measure respecto a Quality para modelo entrenado utilizando sólo los artículos de FayerWayer.

**Fuente:** Elaboración propia.

Como se puede observar en los gráficos anteriores, Quality tiene un gran impacto en la calidad de la clasificación. Cuando se clasifican los datos en forma manual ocurre que muchos artículos no tienen una correlación apreciable con ninguno de los 15 temas encontrados, pues pertenecen a microtemáticas que no alcanzan a ser detectadas por el modelo. En este caso se asocia la clase más cercana dentro de los 15 tópicos como valor real lo que empeora la efectividad de las mediciones. Por otra parte, a este tipo de artículo el clasificador tiende a asociar valores de Quality bajos. Cuando se exige una mayor calidad o nivel de Quality en realidad se acota el conjunto de datos a los más correlacionados con las 15 temáticas encontradas, dejando fuera los artículos que pertenecen a categorías que se encuentran fuera del modelo y por lo tanto no es posible clasificar en forma correcta bajo ninguna circunstancia.

Como ocurre en los análisis anteriores, Quality es un filtro muy estricto, pues a mayor valor, menor cantidad de documentos son analizados como se observa en la figura 5.18.

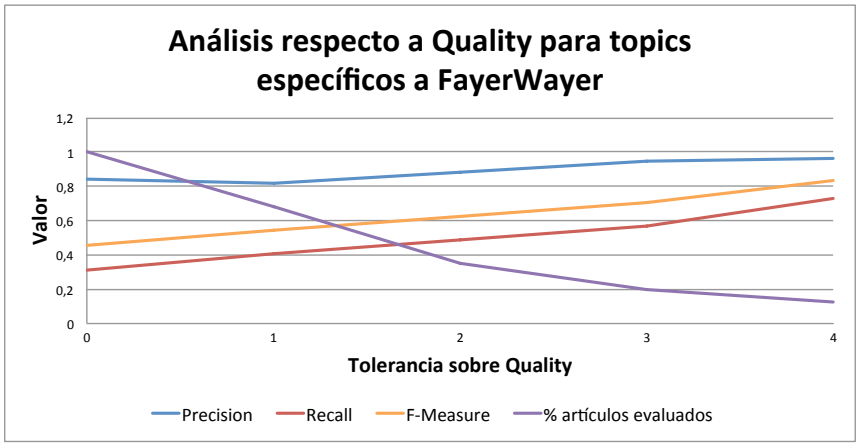


**Figura 5.18:** Análisis de F-Measure, Precision, Recall y el porcentaje de documentos analizados respecto a Quality para modelo entrenado utilizando sólo los artículos de FayerWayer.

**Fuente:** Elaboración propia.

Usualmente un modelo como éste se utilizará anidado dentro de un modelo que incluya a todos los blogs, esto implica que el error de clasificación del primer modelo se propaga al segundo y así sucesivamente. Por ejemplo, si el modelo de ocho blogs posee un 80 % de efectividad, entonces sólo ese 80 % tendrá la posibilidad de ser evaluado correctamente en el modelo específico anidado, lo que indica que si la efectividad del modelo es de un 60 %, por probabilidades condicionales, sólo un 48 %[1] de los datos originales serán clasificados correctamente a nivel de subtemáticas.

### 5.3.3. Análisis de rendimiento

Conocer cómo se comporta el algoritmo de acuerdo a la cantidad de datos a analizar es muy relevante. Actualmente se publican al rededor de 3000 post mensuales entre los 8 blogs analizados de acuerdo a la tabla 5.1 y se espera que con la puesta en marcha del sistema de contenidos generado por los usuarios, la cantidad de artículos aumente en forma considerable.

De acuerdo a Betazeta, el modelo de clasificación debiese ser actualizado por lo menos una vez al mes debido a que las temáticas en algunos blogs varían sustancialmente en el tiempo, como es el caso de sitios asociados a la tecnología o deportes.

El número de operaciones realizadas durante los algoritmos del software siempre dependen de la cantidad de documentos y el número de palabras que cada uno contenga. Si el número de palabras promedio en el conjunto de documentos analizado es $\rho$ y el número de documentos es $\theta$, entonces el orden del software en las etapas de limpieza y entrenamiento estará dada por $O(\theta\rho)$. Por otra parte, para un clasificador que analiza un documento, el número de operaciones dependerá de la cantidad

[1] Este valor corresponde al producto entre 0.6 y 0.8.

de palabras y el número de temáticas que contenga el modelo con el que se clasifica. Si el número de tópicos es $t$ entonces el orden de la clasificación para un documento estará dada por $O(t\rho)$.

1. A partir de los 30.000 documentos obtenidos para pruebas se confecciona un diccionario de palabras con aproximadamente 150.000 términos diferentes.
2. Como en este experimento no interesa la calidad de la predicción, si no que los tiempos de ejecución, utilizando este diccionario se desarrolla un simple algoritmo que toma palabras al azar y genera documentos que contienen entre 200 y 300 palabras.
3. Con lo anterior se generan grupos de 1.000, 10.000, 100.000 y 300.000 documentos. Se seleccionan 300.000 en lugar de 1.000.000 de documentos puesto que el sistema no es capaz de procesar dicha cantidad de datos, mientras que aún es posible medir 300.000.
4. Con estos sets de documentos se mide el tiempo de preparación de datos, el tiempo de entrenamiento y el tiempo que demora el modelo resultante en clasificar un texto de 60, 300 y 1000 palabras.

Las pruebas se ejecutan en un equipo Dell Vostro 3550 con 6 Gb de RAM, Procesador Intel(R) Core(TM) i5-2430M CPU @ 2.40GHz y disco duro 300 Gb HD a 5400 RPM proporcionado por Betazeta, utilizando como servidor web el servidor de desarrollo de Django con sistema operativo Ubuntu 11.10.


**Figura 5.19:** Tiempos de demora del algoritmo al preparar el set de datos y al realizar el entrenamiento para 1.000, 10.000, 100.000 y 300.000 documentos.

**Fuente:** Elaboración propia.

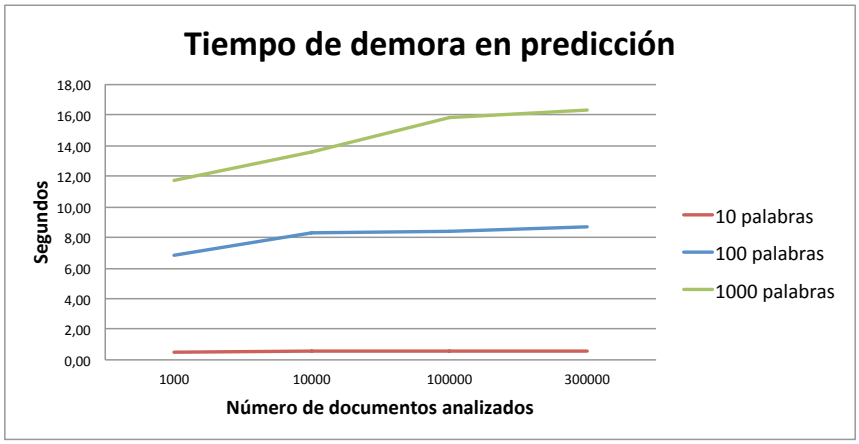


**Figura 5.20:** Tiempos de demora del algoritmo al clasificar un texto de 10, 100 y 1000 palabras usando modelos generados con 1.000, 10.000, 100.000 y 300.000 documentos.

**Fuente:** Elaboración propia.

Como se observa en la figura 5.19 con 300.000 documentos el algoritmo se vuelve mucho más lento. Esto se debe a que el proceso consume toda la memoria RAM del equipo y por tanto se comienza a utilizar memoria virtual causando accesos al disco duro, sin embargo, para las pruebas se utiliza el servidor de desarrollo de Django. Este servidor no está pensado para entornos de producción, ya que es ineficiente manejando la memoria RAM del equipo, además de no soportar conexiones concurrentes. En este sentido Betazeta utilizará servidores pensados para producción como Apache, Nginx u otros, por lo que se espera mejorar el rendimiento al ejecutar los procesos sobre una mejor infraestructura. Por otro lado, una máquina con disco de estado sólido sería capaz de ejectuar el proceso con mayor cantidad de documentos, dado que el acceso a disco en estado sólido es mucho mas veloz.

De acuerdo a lo antes enunciado, actualmente es posible manejar 300.000 documentos con procesos de preparación y entrenamiento que tardan 2.5 horas. Pensando que este proceso debe ser ejecutado una vez al mes, es posible manejar con tranquilidad una tasa de publicación de 1.250 artículos al día en cada uno de los 8 blogs. Tomando en cuenta que hoy la tasa de publicación por blog es de 12,5 artículos diarios en promedio, la plataforma actual puede sostener un crecimiento de por lo menos cien veces la tasa de publicación actual sin optimizar el código de la aplicación, corriendo sobre un notebook común y corriente y utilizando un servidor pensado para desarrollo.

Tomando en cuenta los costos de Amazon EC2, utilizando la configuración *Extra Large*, que tiene un costo de $USD 0.68/hora sobre Linux (ver tabla 3.2), le costaría a Betazeta $USD 17 ejecutar el modelo por lo menos 10 veces al mes con 300.000 documentos.

### Análisis de efectividad aplicando Stemming

Todos los experimentos anteriores no consideran Stemming, esto se debe a que añadir esta fase, por una parte, agrega un costo importante a los tiempos de ejecución del proceso de transformación del texto, por consiguiente se opta por obviarlo en la mayoría de los casos. Por otro lado, un entrenamiento

utilizando Stemming probando diversas configuraciones no permite detectar una buena distribución temática. Por ejemplo, en el caso de un modelo con 8 blogs, no es posible conseguir una distribución que diferencie los 8 blogs, incluso utilizando la mejor configuración encontrada en experimentos anteriores, que corresponde a Alpha 0.1 y Beta 1, sobre los 32.000 documentos de Betazeta.

Este comportamiento se puede explicar debido a que Stemming agrupa palabras en base a un conjunto de reglas gramaticales de manera genérica. Eso provoca que muchas palabras sean llevadas a una raíz incorrecta interfiriendo con la distribución real de las palabras y por ende imposibilitando la obtención de buenos modelos. En particular, el algoritmo de stemming en conjunto con la librería utilizada funcionan muy bien para idiomas como el inglés, pero no es así para el idioma español que presenta una gramática más compleja. Para la implementación de estos experimentos se utiliza la libreria Freeling[2] que permite realizar stemming sobre un texto dado.

[2]http://nlp.lsi.upc.edu/freeling/

# Capítulo 6

# Conclusiones

Luego de presentar la solución y los resultados obtenidos en la experimentación en el desarrollo de los capítulos anteriores, en este capítulo se evalúa el cumplimiento de los objetivos, se describen las dificultades enfrentadas y se realizan propuestas de mejora para el trabajo futuro que enmarca el proyecto.

## 6.1. Evaluación

De acuerdo a lo descrito en esta memoria, Tanalyzer como prototipo de plataforma de análisis de texto, permite validar la hipótesis inicial sobre la factibilidad de construir una herramienta que permita visualizar la distribución de contenidos y clasificar un texto plano en el contexto de la empresa.

Respecto a la visualización de la información se puede mencionar que con la herramienta es posible "Desentrañar" el conocimiento implícito que se encuentra en los Blogs, visualizando en qué forma se distribuyen los temas dentro de los artículos escritos, sin la necesidad de revisarlos en forma individual, esto es fundamental para saber qué tipo de contenido es necesario potenciar o descontinuar de las líneas editoriales según lo indique el mercado. Si bien éste método depende del modelo y el conjunto de datos, es un primer acercamiento a una solución robusta.

Asímismo, la clasificación de texto se demuestra posible en base a los resultados obtenidos. Si bien *F-Measure* en el caso del modelo específico a FayerWayer no es tan alto como el modelo de 8 temáticas, *Precision* sí obtuvo resultados aceptables. Esto implica que si bien la respuesta final que se le entregará al usuario tendrá una tendencia a omitir resultados correctos, las respuestas sí serán precisas, lo cual se encuentra en un rango aceptable en esta etapa de prototipo. Por otra parte, el modelo que permite identificar a qué blog pertenece un texto, es decir el modelo de temáticas generales, obtuvo buenos resultados en ambos indicadores.

En referencia a las etapas de KDD y las metodologías de Data Mining, para poder diseñar e implementar el software, fue absolutamente necesario conocer la empresa, el contexto del problema, los procesos internos del negocio, visualizar y comprender el formato de los datos, ejecutar las transformaciones necesarias para el proceso de extracción, y definir el tamaño de las muestras a evaluar para cada blog, así como comprender el contenido de éstos.

Inicialmente el sistema se construyó como aplicación uni-modelo, es decir, se entrenaba un modelo con una configuración determinada, el cual se esperaba utilizar en todos los casos posibles. Trabajar con metodologías ágiles, siguiendo un patrón de desarrollo iterativo que genera valor en cada fase, permitió dilucidar en las etapas iniciales de este proyecto que una plataforma uni-modelo resultaría poco efectiva. Los experimentos expuestos en este trabajo así lo demuestran puesto que se observa una clara disminución de la efectividad de un modelo para predecir si el número de clases disponibles para clasificación aumenta, en particular, disminuye el *Recall*. Con un sistema uni-modelo habría sido necesario detectar por lo menos 80 clases distintas para poder trabajar con los 8 blogs en estudio.

Las métricas definidas para la evaluación permiten medir con éxito la capacidad de un modelo predictor. Cuando *F-Measure* toma un valor de 1 entonces se tiene que la clasificación no contiene errores, esto implica que no hay falsos positivos ni negativos.

La investigación teórica respecto a los modelos existentes, en partícular sobre LDA, permiten definir las líneas para comenzar a desarrollar una plataforma en torno a este modelo de clasificación, y posteriormente interpretar sus resultados.

En resumen, cada uno de los objetivos específicos planteados inicialmente se cumplen a cabalidad, así como también el objetivo general de este trabajo. Enumerándolos de acuerdo a lo recién expuesto se tiene que:

1. Se entendió la problemática que involucra a la empresa, tanto a nivel de negocio como técnico. Por otro lado la investigación realizada, que convergió en el marco teórico, permitió adquirir los conocimientos necesarios en el área de estudio.
2. A partir de lo anterior se definió LDA como el modelo central y *Precision*, *Recall* y *F-Measure* como medidas de efectividad.
3. En base a la investigación antes mencionada y sumada a la experiencia de las primeras iteraciones del software, se diseña un sistema multimodelo para poder visualizar información y clasificar documentos.
4. Utilizando el conocimiento adquirido y los modelos planteados se desarrolló la plataforma de clasificación y visualización de contenidos.
5. Finalmente, se evaluó el software para medir sus resultados, concluyendo que la solución es viable en el contexto de la empresa y la etapa del proyecto.

De los puntos enumerados anteriormente se desprende que se construyó con éxito una plataforma de evaluación y visualización que permitirán apoyar a Betazeta en el futuro, cuando se ponga en marcha el sistema de generación de contenidos a partir de los usuarios de la red de blogs, lo que implica el cumplimiento del objetivo general de ésta memoria.

## 6.2. Dificultades Encontradas

A continuación se comentan algunas de los desafíos que se debieron resolver a lo largo de este proyecto.

### 6.2.1. Codificación de caracteres

Uno de los primeros desafíos encontrados consistió en manipular los sistemas de codificación del lenguaje Python para poder hacer una correcta extracción de los artículos de Betazeta mediante scrapping a causa de que no se contó con acceso a las bases de datos hasta muy entrado el desarrollo del proyecto. El mecanismo que usa el lenguaje es complejo, debido a que es muy específico, por lo tanto cambios leves en algún formato de los caracteres inmediatamente genera errores. Este problema es particularmente frecuente al trabajar con idioma español, debido al uso de la letra ñ y tildes. Por otro lado, librerías externas como Freeling usualmente manejan su propio tipo de codificación, generando diversas inconsistencias en la transformación de textos.

### 6.2.2. El sistema multimodelo

Diseñar el sistema multimodelo anidado supuso un desafío en sí mismo a causa de que al observar la inviabilidad de una plataforma uni-modelo fue necesario encontrar un método para resolver la clasificación.

La complejidad del sistema multimodelo radica en que es necesario hacer un manejo consistente de todas las entidades involucradas, como por ejemplo, los resultados de los entrenamientos, el set de palabras característico de cada modelo o la medición de diferentes distribuciones de palabras para detectar qué términos son relevantes o no dentro de un modelo específico, entre otros múltiples factores que pueden inducir a error.

Al trabajar con muchos modelos de clasificación el modelo entidad relación crece y las relaciones se vuelven más complejas. Diseñar una estructura que permita manejar el alto volumen de datos que se genera y acumulan continuamente en el sistema, del orden de gigabytes de datos, al entrenar un conjunto de textos en este contexto presenta múltiples desafíos respecto a la optimización de consultas, implementación de buenos índices, manejo de diversos cachés a lo largo de la aplicación, incorporación de threads o hilos para ejecutar procesos en paralelo, etc.

## 6.3. Trabajo Futuro

De acuerdo a los análisis realizados el prototipo presentado en este trabajo cumple con las expectativas planteadas inicialmente, pero aún existe un gran espacio para mejoras que debe realizar la empresa para poder llevar el proyecto a producción con mejores resultados.

### 6.3.1. Integración de modelos

En primer lugar, el desarrollo de la aplicación se centra en LDA, utilizándolo como único modelo disponible. De acuerdo a la literatura existen múltiples variaciones de este método, así como también otros mecanismos que permiten generar clasificación no supervisada. Es interesante incorporar un modelo que tome en cuenta variables como la temporalidad de las temáticas y permita ir evolucionando un topic conforme transcurre el tiempo [34].

Por ejemplo, hoy en día cuando hablamos de tecnología y redes sociales, palabras como facebook,

twitter, iphone o android tienen gran peso, pero esto no siempre será así, porque a medida que las tecnologías cambien, los términos asociados también lo harán. Un ejemplo de esto se puede observar en los últimos meses con el sitio http://pinterest.com, que ha tomado una gran relevancia en el área, es esperable que sitios emergentes como éste alteren la estructura de palabras de ciertos tópicos con el tiempo. Actualmente es posible abordar este problema entrenando nuevos modelos y realizando nuevas clasificaciones pero se pierde la historia anterior de un topic, ya que cada temática se asocia a un modelo que deja de estar vigente, por ende definir que dos temáticas de diferentes modelos son la misma en distintos puntos del tiempo es una tarea que debe realizarse manualmente. Una interfaz para esto último tampoco esta implementada y podría plantearse como desafío futuro.

Continuando con los modelos predictivos, al ser LDA un modelo no supervisado proporciona ventajas, pues hoy no se tiene la capacidad de asignar categorías en forma manual, pero presenta el problema de que las temáticas no son controladas, sino que son dadas por los textos y la configuración del modelo. Es bueno tener mayor control sobre el proceso de entrenamiento utilizando un sistema de topic modeling supervisado [5] una vez esté en marcha el sistema de contenidos generados por el usuario, es posible utilizar a los mismos usuarios para asignar una clasificación a sus propios documentos, disminuyendo el costo de la clasificación inicial que alimente el modelo supervisado.

### 6.3.2. Funcionalidades

Es posible agregar funcionalidades que permitan realizar una evaluación de modelos más amigable. Actualmente existen rutinas que hacen los cálculos de *Precision*, *Recall* y *F-Measure*, pero fueron diseñadas para el uso interno en la evaluación de ésta memoria y no como parte del software a nivel de usuario, ya que su output se imprime en el terminal desde donde se ejecuta el servidor de Django y luego estos datos se trabajan en Microsoft Excel, por lo tanto, como desafío futuro, se piensa en la posibilidad de integrar todo el proceso de evaluación dentro de la plataforma y obtener las estádisticas directamente sin tener que usar un software externo.

Por otro lado es útil agregar más parámetros configurables, como el número de iteraciones a realizar por un modelo así como también integrar la posibilidad de incluir expresiones regulares a en la limpieza de texto para modificar los documentos en forma personalizada.

Si bien dentro de Tanalyzer existe el módulo que permite clasificar un texto plano, seleccionar un modelo y obtener una clasificación, es deseable diseñar un prototipo de interfaz que esté centrado de lleno en el usuario final de la plataforma, los futuros generadores de contenidos externos a Betazeta. En este sentido, construir una interfaz que permita al usuario redactar un documento y que mientras lo escribe aparezcan sugerencias que indiquen en qué área clasificar el documento, se presenten las keywords más relevantes dentro del texto o se señale que el Post escrito no es posible ser clasificado en las áreas definidas en forma automática, todo lo anterior en tiempo real, no es una tarea compleja utilizando el webservice disponible en Tanalyzer, además de que agrega mucho valor a la plataforma desde el punto de vista de la usabilidad y la interacción con el autor de un nuevo documento.

### 6.3.3. Optimizaciones al software y el proceso de desarrollo

Como se menciona en el Capítulo 5, existen posibilidades para realizar optimizaciones tanto en la plataforma como en los procesos de preparación y entrenamiento de datos y clasificación. Se puede mejorar la utilización de memoria RAM instalando un equipo como servidor dedicado, mejorando el rendimiendo en los procesos con un alto volumen de documentos al evitar el acceso a disco.

Por otro lado, respecto al software, existen varias mejoras realizables al prototipo construido. En primer lugar, se debe trabajar para mejorar el bajo *Recall* obtenido cuando se aumenta la cantidad de modelos.

Respecto a la eficiencia del software, modificar la implementación para aprovechar mejor la capacidad de procesadores con más de un núcleo permitiría disminuir los tiempos de ejecución de los algoritmos y la clasificación de documentos. Además, es posible hacer mejoras en la eficiencia de los algoritmos, diseñar un mejor sistema de caché para una clasificación más veloz, así como también optimizar las consultas y disminuir la cantidad de conexiones que se realizan a la base de datos. También es posible trasladar rutinas pesadas a lenguajes de bajo nivel que permitan una ejecución más rápida, como C.

También es deseable incorporar test unitarios para validar los procesos en forma consistente. Esto presenta una alta complejidad, dada la cantidad de datos involucrados para poder validar ciertas componentes del software. Actualmente existe una implementación parcial de test unitarios que permiten revisar algunos aspectos en la ejecución de procesos, pero se encuentra incompleta, ya que sólo revisan parte del software. Esto no se documenta en el desarrollo del trabajo debido a que escapa a los alcances de este informe pues requiere una explicación extensa del método que utiliza Django para la implementación de test unitarios y las distintas herramientas que provee, lo que se relaciona sólo indirectamente con el objetivo principal de esta memoria.

### 6.3.4. Liberación del software desarrollado

Finalmente, la propiedad intelectual de Tanalyzer pertenece al autor de ésta memoria. En este contexto el código fuente de Tanalyzer está disponible en el sitio GitHub en la url:

*https://github.com/decklord/Tanalyzer/*

como proyecto de código abierto sobre el cual se espera seguir trabajando y a la vez recibir contribuciones de terceros para poder llevar a cabo las mejoras antes mencionadas.

# Bibliografía


[1] Loulwah Alsumait, Pu Wang, Carlotta Domeniconi, and Daniel Barbará. *Embedding Semantics in LDA Topic Models*, pages 183–204. John Wiley & Sons, Ltd, 2010.

[2] D. Andrzejewski, A. Mulhern, B. Liblit, and X. Zhu. Statistical debugging using latent topic models. *Machine Learning: ECML 2007*, pages 6–17, 2007.

[3] D. Blei. Introduction to probabilistic topic models. *Communications of the ACM*, 2011.

[4] D. Blei and J. Lafferty. Dynamic topic models. *International Conference on Machine Learning*, pages 113–120, 2006.

[5] D.M. Blei and J.D. McAuliffe. Supervised topic models. *Arxiv preprint arXiv:1003.0783*, 2010.

[6] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[7] W.B. Cavnar and J.M. Trenkle. N-gram-based text categorization. *Ann Arbor MI*, 48113:4001, 1994.

[8] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rudiger Wirth. Crisp-dm 1.0 step-by-step data mining guide. Technical report, The CRISP-DM consortium, August 2000.

[9] A. Cockburn and J. Highsmith. Agile software development, the people factor. *Computer Magazine*, 34(11):131–133, 2001.

[10] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science, 41(6):391–407*, 1990.

[11] Raquel Gómez Díaz. *Estudio de la incidencia del conocimiento lingüístico en los sistemas de recuperación de la información para el español*. PhD thesis, Universidad de Salamanca, 2001.

[12] The Economist. Blogging is just another word for having conversations. Web, Abril 2006.

[13] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[14] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, 1996.

[15] G. Figuerola, Á.F. Zazo, and J.L. Alonso-Berrocal. La recuperación de información en espanol y la normalización de términos. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, 22:135–145, 2004.

[16] N. Guduru. Text mining with support vector machines and non-negative matrix factorization algorithms. Master's thesis, University of Rhode Island, 2006.

[17] J. Highsmith and A. Cockburn. Agile software development: The business of innovation. *Computer*, 34(9):120–127, 2001.

[18] T. Hoffman. Probabilistic latent semantic analysis. *Uncertainty in Artificial Intelligence (UAI)*, 1999.

[19] A.G.K. Janecek and W.N. Gansterer. Utilizing nonnegative matrix factorization for email classification problems. In *Text Mining: Applications and Theory*, pages 57–80. Wiley Online Library, 2010.

[20] Eric P. Jiang. Content-based spam email classification using machine-learning algorithms. In *Text Mining: Applications and Theory (eds M. W. Berry and J. Kogan)*, chapter 3. John Wiley & Sons, Ltd, Chichester, UK., 2010.

[21] April Kontostathis, Lynne Edwards, and Amanda Leatherman. Text mining and cybercrime. In *Text Mining, Applications and Theory*, chapter 8. John Wiley & Sons, Ltd, Chichester, UK., 2010.

[22] Daniel T. Larose. *Discovering Knowledge in Data: An Introduction to Data Mining*. John Wiley & Sons, Inc., 2005.

[23] Seyyed Soroush Rohanizadeha; Mohammad Bameni Moghadama. A proposed data mining methodology and its application to industrial procedures. *Journal of Industrial Engineering 4*, pages 37–50, 2009.

[24] Andrew McCallum;Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48, 1998.

[25] Bryce Glass Randy Farmer. *Building Web Reputation Systems*. O'Reilly Media / Yahoo Press, 2010.

[26] T. Reenskaug. The model-view-controller (mvc) its past and present. *University of Oslo Draft*, 2003.

[27] D. Riehle. *Framework design: A role modeling approach*. PhD thesis, Zürich, 2000.

[28] S. Rose, D. Engel, N. Cramer, and W. Cowley. Automatic keyword extraction from individual documents. In *Text Mining: Applications and Theory*, pages 1–20. Wiley Online Library, 2010.

[29] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, November 1975.

[30] M. Steyvers and T. Griffiths. Probabilistic topic models. *Handbook of latent semantic analysis*, 427(7):424–440, 2007.

[31] Y.W. Teh, M.I. Jordan, M.J. Beal, and D.M. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[32] J.D. Velásquez and V. Palade. Adaptive web sitesa knowledge extraction from web data approach. In *Proceeding of the 2008 conference on Adaptive Web Sites: A Knowledge Extraction from Web Data Approach*, pages 1–272. IOS Press, 2008.

[33] H.M. Wallach. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984. ACM, 2006.

[34] X. Wang and A. McCallum. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433. ACM, 2006.

[35] S.M. Weiss. *Text mining: predictive methods for analyzing unstructured information*. Springer-Verlag New York Inc, 2005.

[36] W.J. Wilbur and K. Sirotkin. The automatic identification of stop words. *Journal of information science*, 18(1):45, 1992.

# Apéndices

## A . Listado de Stopwords

A continuación se presenta el listado de Stopwords utilizado.

| Stop Words | | | | | | |
|---|---|---|---|---|---|---|
| a | aca | acá | ademas | además | ahi | ahora |
| ahí | ajena | ajenas | ajeno | ajenos | al | algo |
| algun | alguna | algunas | alguno | algunos | algún | alla |
| alli | allí | ambos | ampleamos | ante | antes | aquel |
| aquella | aquellas | aquello | aquellos | aqui | aquí | arriba |
| asi | así | atras | aun | aunque | años | b |
| bajo | bastante | betazeta | bien | buena | bueno | c |
| cabe | cada | casa | casi | caso | cc | chile |
| cierta | ciertas | cierto | ciertos | cinco | claro | com |
| como | compañía | con | conmigo | conseguimos | conseguir | consigo |
| consigue | consiguen | consigues | contigo | contra | cosas | cual |
| cuales | cualquier | cualquiera | cualquieras | cuan | cuando | cuanta |
| cuantas | cuanto | cuantos | cuatro | cuenta | cuál | cuáles |
| cómo | d | da | daba | dar | de | decir |
| dejar | del | demas | demasiada | demasiadas | demasiado | demasiados |
| demás | dentro | desde | después | dia | dias | diez |
| dio | disponible | dió | donde | dos | durante | día |
| días | e | el | ella | ellas | ello | ellos |
| embargo | empleais | emplean | emplear | empleas | empleo | en |
| encima | entonces | entre | equipo | era | erais | eramos |
| eran | eras | eres | es | esa | esas | ese |
| eso | esos | esta | estaba | estabais | estabamos | estaban |
| estabas | estad | estada | estadas | estado | estados | estais |
| estamos | están | estando | estar | estara | estaran | estaras |
| estare | estareis | estaremos | estaria | estariais | estariamos | estarian |

| Stop Words | | | | | | |
|---|---|---|---|---|---|---|
| estarias | estará | estarán | estarás | estaré | estaréis | estaría |
| estaríais | estaríamos | estarían | estarías | estas | este | esteis |
| estemos | esten | estes | esto | estos | estoy | estuve |
| estuviera | estuvierais | estuvieramos | estuvieran | estuvieras | estuvieron | estuviese |
| estuvieseis | estuviesemos | estuviesen | estuvieses | estuvimos | estuviste | estuvisteis |
| estuviérais | estuviéramos | estuviésemos | estuvo | estás | esté | estéis |
| estén | estés | etc | f | fin | final | forma |
| fue | fuente | fuera | fuerais | fueramos | fueran | fueras |
| fueron | fuese | fueseis | fuesemos | fuesen | fueses | fui |
| fuimos | fuiste | fuisteis | fué | fuérais | fuéramos | fuéseis |
| fuésemos | fuí | g | galeria | gente | gracias | gran |
| grandes | grupo | gueno | h | ha | habeis | habia |
| habiais | habiamos | habian | habias | habida | habidas | habido |
| habidos | habiendo | habra | habran | habras | habre | habreis |
| habremos | habria | habriais | habriamos | habrian | habrias | habrá |
| habrán | habrás | habré | habréis | habría | habríais | habríamos |
| habrían | habrías | había | habíais | habíamos | habían | habías |
| hace | haceis | hacemos | hacen | hacer | haces | hacia |
| hacía | hago | han | has | hasta | hay | haya |
| hayais | hayamos | hayan | hayas | he | hecho | hemos |
| historia | horas | hoy | http | hube | hubiera | hubierais |
| hubieramos | hubieran | hubieras | hubieron | hubiese | hubieseis | hubiesemos |
| hubiesen | hubieses | hubimos | hubiste | hubisteis | hubiérais | hubiéramos |
| hubiésemos | hubo | i | idea | importante | incluso | incluye |
| informacion | información | intenta | intentais | intentamos | intentan | intentar |
| intentas | intento | intentó | ir | j | jamas | jamás |
| juego | junto | juntos | k | l | la | lado |
| largo | las | le | les | link | lo | los |
| luego | lugar | m | manera | mas | mayor | me |
| medio | mejor | menos | mercado | meses | mi | mia |
| mias | mientras | millones | minuto | minutos | mio | mios |
| mis | misma | mismas | mismo | mismos | modelo | modo |
| momento | mucha | muchas | muchisima | muchisimas | muchisimo | muchisimos |
| mucho | muchos | muchísima | muchísimas | muchísimo | muchísimos | mundo |
| muy | más | mía | mías | mío | míos | nada |
| ni | ningun | ninguna | ningunas | ninguno | ningunos | ningún |
| no | nombre | nos | nosotras | nosotros | nuestra | nuestras |
| nuestro | nuestros | nueva | nueve | nuevo | nunca | o |
| ocho | os | otra | otras | otro | otros | p |
| para | parece | parecer | parte | pasado | paso | país |
| pero | personas | poca | pocas | poco | pocos | podeis |
| podemos | poder | podria | podriais | podriamos | podrian | podrias |
| podría | podríais | podríamos | podrían | podrías | por | por que |
| por qué | porque | porqué | primer | primera | primero | primero desde |
| pronto | proximo | próximo | publicado | puede | pueden | puedes |

| Stop Words | | | | | | |
|---|---|---|---|---|---|---|
| puedo | pues | q | que | querer | quien | quienes |
| quienesquiera | quienquiera | quiza | quizas | quizá | quizás | quién |
| quiénes | qué | r | s | sabe | sabeis | sabemos |
| saben | saber | sabes | sabéis | se | sea | seais |
| seamos | sean | seas | segun | segundo | segundos | según |
| seis | semana | ser | sera | seran | seras | sere |
| sereis | seremos | seria | seriais | seriamos | serian | serias |
| será | serán | serás | seré | seréis | sería | seríais |
| seríamos | serían | serías | si | sido | siempre | siendo |
| siete | sin | sino | sistema | so | sobre | sois |
| solamente | solo | somos | son | soy | sr | sra |
| sres | srta | sta | su | sus | suya | suyas |
| suyo | suyos | sé | sí | t | tal | tales |
| tambien | también | tampoco | tan | tanta | tantas | tanto |
| tantos | te | tendra | tendran | tendras | tendre | tendreis |
| tendremos | tendria | tendriais | tendriamos | tendrian | tendrias | tendrá |
| tendrán | tendrás | tendré | tendréis | tendría | tendríais | tendríamos |
| tendrían | tendrías | tened | teneis | tenemos | tener | tenga |
| tengais | tengamos | tengan | tengas | tengo | tenia | teniais |
| teniamos | tenian | tenias | tenida | tenidas | tenido | tenidos |
| teniendo | tenía | teníais | teníamos | tenían | tenías | the |
| ti | tiempo | tiene | tienen | tienes | tipo | toda |
| todas | todavia | todavía | todo | todos | tomar | trabaja |
| trabajais | trabajamos | trabajan | trabajar | trabajas | trabajo | trabajáis |
| trabajó | tras | trata | traves | través | tres | tu |
| tus | tuve | tuviera | tuvierais | tuvieramos | tuvieran | tuvieras |
| tuvieron | tuviese | tuvieseis | tuviesemos | tuviesen | tuvieses | tuvimos |
| tuviste | tuvisteis | tuviérais | tuviéramos | tuviésemos | tuvo | tuya |
| tuyas | tuyo | tuyos | té | tú | u | ultimo |
| un | una | unas | uno | unos | usa | usais |
| usamos | usan | usar | usas | uso | usted | ustedes |
| v | va | vais | valor | vamos | van | varias |
| varios | vaya | veces | ver | verdad | verdadera | version |
| versión | vez | via | vida | video | vosotras | vosotros |
| voy | vuestra | vuestras | vuestro | vuestros | vía | w |
| www | x | y | ya | yo | z | él |

# B . Tablas de Datos

En este apartado se encuentran los datos a partir de los cuales se obtienen los gráficos del Capítulo 5.

| Precision | | | | | |
|---|---|---|---|---|---|
| **Tolerance** | **Alpha 0,5** | **Alpha 0,4** | **Alpha 0,3** | **Alpha 0,1** | **Alpha 0,09** |
| 1 | 0.1266 | 0.1433 | 0.1200 | 0.1404 | 0.1431 |
| 2 | 0.1437 | 0.1612 | 0.1270 | 0.1553 | 0.1565 |
| 3 | 0.1650 | 0.1852 | 0.1420 | 0.1764 | 0.1776 |
| 4 | 0.1928 | 0.2170 | 0.1652 | 0.2040 | 0.2049 |
| 5 | 0.2262 | 0.2579 | 0.1923 | 0.2420 | 0.2409 |
| 6 | 0.2683 | 0.3024 | 0.2280 | 0.2862 | 0.2867 |
| 7 | 0.3113 | 0.3569 | 0.2727 | 0.3395 | 0.3388 |
| 8 | 0.3629 | 0.4262 | 0.3341 | 0.4063 | 0.4066 |
| 9 | 0.3629 | 0.4262 | 0.3341 | 0.4063 | 0.4066 |
| 10 | 0.3629 | 0.4262 | 0.3341 | 0.4063 | 0.4066 |
| 11 | 0.3945 | 0.4771 | 0.3803 | 0.4508 | 0.4543 |
| 12 | 0.4345 | 0.5209 | 0.4228 | 0.4974 | 0.4914 |
| 13 | 0.4660 | 0.5587 | 0.4668 | 0.5455 | 0.5337 |
| 14 | 0.5065 | 0.5962 | 0.5087 | 0.5930 | 0.5805 |
| 15 | 0.5406 | 0.6260 | 0.5478 | 0.6454 | 0.6162 |
| 16 | 0.5718 | 0.6573 | 0.5871 | 0.6872 | 0.6453 |
| 17 | 0.6019 | 0.6926 | 0.6250 | 0.7393 | 0.6787 |
| 18 | 0.6249 | 0.7250 | 0.6607 | 0.7697 | 0.7108 |
| 19 | 0.6249 | 0.7250 | 0.6607 | 0.7697 | 0.7108 |
| 20 | 0.6249 | 0.7250 | 0.6607 | 0.7697 | 0.7108 |
| 21 | 0.6470 | 0.7435 | 0.6887 | 0.8023 | 0.7328 |
| 22 | 0.6631 | 0.7574 | 0.7068 | 0.8200 | 0.7539 |
| 23 | 0.6726 | 0.7714 | 0.7283 | 0.8403 | 0.7677 |
| 24 | 0.6883 | 0.7801 | 0.7560 | 0.8502 | 0.7731 |
| 25 | 0.6930 | 0.7847 | 0.7733 | 0.8691 | 0.7797 |
| 26 | 0.7061 | 0.7977 | 0.7940 | 0.8802 | 0.8103 |
| 27 | 0.7123 | 0.8055 | 0.8089 | 0.9023 | 0.8250 |
| 28 | 0.7269 | 0.8130 | 0.8196 | 0.9080 | 0.8370 |
| 29 | 0.7269 | 0.8130 | 0.8196 | 0.9080 | 0.8370 |
| 30 | 0.7269 | 0.8130 | 0.8196 | 0.9080 | 0.8370 |
| 31 | 0.7319 | 0.8182 | 0.8224 | 0.9155 | 0.8564 |
| 32 | 0.7422 | 0.8280 | 0.8381 | 0.9269 | 0.8638 |
| 33 | 0.7454 | 0.8401 | 0.8559 | 0.9348 | 0.8741 |
| 34 | 0.7430 | 0.8428 | 0.8665 | 0.9348 | 0.8898 |

**Cuadro 6.1:** Precision para un entrenamiento utilizando Beta 1, genera la figura 5.1.

| Recall | | | | | |
|---|---|---|---|---|---|
| **Tolerance** | **Alpha 0,5** | **Alpha 0,4** | **Alpha 0,3** | **Alpha 0,1** | **Alpha 0,09** |
| 1 | 0.8251 | 0.8039 | 0.7923 | 0.7971 | 0.8223 |
| 2 | 0.8240 | 0.8039 | 0.7923 | 0.7971 | 0.8213 |
| 3 | 0.8177 | 0.7986 | 0.7902 | 0.7961 | 0.8213 |
| 4 | 0.8072 | 0.7944 | 0.7892 | 0.7908 | 0.8181 |
| 5 | 0.7987 | 0.7902 | 0.7829 | 0.7876 | 0.8170 |
| 6 | 0.7893 | 0.7755 | 0.7723 | 0.7823 | 0.8107 |
| 7 | 0.7765 | 0.7692 | 0.7639 | 0.7769 | 0.8042 |
| 8 | 0.7650 | 0.7586 | 0.7555 | 0.7674 | 0.7905 |
| 9 | 0.7650 | 0.7586 | 0.7555 | 0.7674 | 0.7905 |
| 10 | 0.7650 | 0.7586 | 0.7555 | 0.7674 | 0.7905 |
| 11 | 0.7461 | 0.7523 | 0.7387 | 0.7569 | 0.7789 |
| 12 | 0.7364 | 0.7406 | 0.7241 | 0.7430 | 0.7598 |
| 13 | 0.7216 | 0.7258 | 0.7102 | 0.7331 | 0.7404 |
| 14 | 0.7090 | 0.7131 | 0.6996 | 0.7162 | 0.7278 |
| 15 | 0.6941 | 0.6974 | 0.6891 | 0.6983 | 0.7120 |
| 16 | 0.6813 | 0.6869 | 0.6766 | 0.6802 | 0.6936 |
| 17 | 0.6718 | 0.6645 | 0.6682 | 0.6706 | 0.6713 |
| 18 | 0.6465 | 0.6519 | 0.6502 | 0.6545 | 0.6598 |
| 19 | 0.6465 | 0.6519 | 0.6502 | 0.6545 | 0.6598 |
| 20 | 0.6465 | 0.6519 | 0.6502 | 0.6545 | 0.6598 |
| 21 | 0.6284 | 0.6299 | 0.6343 | 0.6429 | 0.6449 |
| 22 | 0.6200 | 0.6182 | 0.6164 | 0.6302 | 0.6228 |
| 23 | 0.6029 | 0.6098 | 0.5995 | 0.6187 | 0.6102 |
| 24 | 0.5889 | 0.5970 | 0.5856 | 0.6026 | 0.5834 |
| 25 | 0.5741 | 0.5727 | 0.5581 | 0.5888 | 0.5685 |
| 26 | 0.5549 | 0.5514 | 0.5411 | 0.5622 | 0.5559 |
| 27 | 0.5422 | 0.5409 | 0.5251 | 0.5506 | 0.5358 |
| 28 | 0.5305 | 0.5207 | 0.5050 | 0.5306 | 0.5185 |
| 29 | 0.5305 | 0.5207 | 0.5050 | 0.5306 | 0.5185 |
| 30 | 0.5305 | 0.5207 | 0.5050 | 0.5306 | 0.5185 |
| 31 | 0.5122 | 0.5089 | 0.4836 | 0.5095 | 0.5067 |
| 32 | 0.4996 | 0.4962 | 0.4710 | 0.4927 | 0.4886 |
| 33 | 0.4795 | 0.4773 | 0.4593 | 0.4790 | 0.4738 |
| 34 | 0.4604 | 0.4581 | 0.4422 | 0.4664 | 0.4590 |

**Cuadro 6.2:** Recall para un entrenamiento utilizando Beta 1, genera la figura 5.2.

| F-Measure | | | | | |
|---|---|---|---|---|---|
| **Tolerance** | **Alpha 0,5** | **Alpha 0,4** | **Alpha 0,3** | **Alpha 0,1** | **Alpha 0,09** |
| 1 | 0.2196 | 0.2432 | 0.2084 | 0.2387 | 0.2438 |
| 2 | 0.2447 | 0.2685 | 0.2189 | 0.2599 | 0.2628 |
| 3 | 0.2746 | 0.3006 | 0.2407 | 0.2888 | 0.2921 |
| 4 | 0.3112 | 0.3409 | 0.2733 | 0.3243 | 0.3277 |
| 5 | 0.3526 | 0.3889 | 0.3087 | 0.3702 | 0.3721 |
| 6 | 0.4004 | 0.4352 | 0.3520 | 0.4190 | 0.4236 |
| 7 | 0.4445 | 0.4876 | 0.4019 | 0.4725 | 0.4767 |
| 8 | 0.4922 | 0.5458 | 0.4633 | 0.5313 | 0.5370 |
| 9 | 0.4922 | 0.5458 | 0.4633 | 0.5313 | 0.5370 |
| 10 | 0.4922 | 0.5458 | 0.4633 | 0.5313 | 0.5370 |
| 11 | 0.5161 | 0.5839 | 0.5021 | 0.5651 | 0.5738 |
| 12 | 0.5465 | 0.6116 | 0.5338 | 0.5959 | 0.5968 |
| 13 | 0.5663 | 0.6314 | 0.5633 | 0.6255 | 0.6203 |
| 14 | 0.5909 | 0.6494 | 0.5891 | 0.6488 | 0.6459 |
| 15 | 0.6078 | 0.6598 | 0.6104 | 0.6708 | 0.6606 |
| 16 | 0.6217 | 0.6718 | 0.6287 | 0.6837 | 0.6686 |
| 17 | 0.6350 | 0.6782 | 0.6459 | 0.7033 | 0.6750 |
| 18 | 0.6355 | 0.6865 | 0.6554 | 0.7074 | 0.6843 |
| 19 | 0.6355 | 0.6865 | 0.6554 | 0.7074 | 0.6843 |
| 20 | 0.6355 | 0.6865 | 0.6554 | 0.7074 | 0.6843 |
| 21 | 0.6376 | 0.6820 | 0.6604 | 0.7138 | 0.6861 |
| 22 | 0.6408 | 0.6808 | 0.6585 | 0.7127 | 0.6821 |
| 23 | 0.6358 | 0.6811 | 0.6576 | 0.7127 | 0.6799 |
| 24 | 0.6347 | 0.6763 | 0.6599 | 0.7053 | 0.6650 |
| 25 | 0.6280 | 0.6621 | 0.6483 | 0.7020 | 0.6575 |
| 26 | 0.6214 | 0.6520 | 0.6436 | 0.6861 | 0.6594 |
| 27 | 0.6157 | 0.6472 | 0.6368 | 0.6839 | 0.6497 |
| 28 | 0.6134 | 0.6349 | 0.6249 | 0.6698 | 0.6403 |
| 29 | 0.6134 | 0.6349 | 0.6249 | 0.6698 | 0.6403 |
| 30 | 0.6134 | 0.6349 | 0.6249 | 0.6698 | 0.6403 |
| 31 | 0.6026 | 0.6275 | 0.6091 | 0.6547 | 0.6367 |
| 32 | 0.5972 | 0.6205 | 0.6031 | 0.6434 | 0.6242 |
| 33 | 0.5836 | 0.6087 | 0.5978 | 0.6335 | 0.6145 |
| 34 | 0.5685 | 0.5936 | 0.5856 | 0.6223 | 0.6056 |

**Cuadro 6.3:** F-Measure para un entrenamiento utilizando Beta 1, genera la figura 5.3.

| Recall | | | | | | |
|---|---|---|---|---|---|---|
| **Tolerance** | **Beta 2** | **Beta 1** | **Beta 0,9** | **Beta 0,7** | **Beta 0,5** | **Beta 0,4** |
| 1 | 0.8132 | 0.7971 | 0.8072 | 0.7863 | 0.7989 | 0.7943 |
| 2 | 0.8100 | 0.7971 | 0.8072 | 0.7863 | 0.7978 | 0.7932 |
| 3 | 0.8005 | 0.7961 | 0.8051 | 0.7852 | 0.7978 | 0.7922 |
| 4 | 0.7825 | 0.7908 | 0.7967 | 0.7821 | 0.7936 | 0.7849 |
| 5 | 0.7751 | 0.7876 | 0.7904 | 0.7779 | 0.7873 | 0.7754 |
| 6 | 0.7656 | 0.7823 | 0.7789 | 0.7694 | 0.7831 | 0.7639 |
| 7 | 0.7529 | 0.7769 | 0.7737 | 0.7620 | 0.7767 | 0.7522 |
| 8 | 0.7369 | 0.7674 | 0.7610 | 0.7535 | 0.7598 | 0.7407 |
| 9 | 0.7369 | 0.7674 | 0.7610 | 0.7535 | 0.7598 | 0.7407 |
| 10 | 0.7369 | 0.7674 | 0.7610 | 0.7535 | 0.7598 | 0.7407 |
| 11 | 0.7264 | 0.7569 | 0.7505 | 0.7399 | 0.7430 | 0.7239 |
| 12 | 0.7157 | 0.7430 | 0.7336 | 0.7178 | 0.7220 | 0.7145 |
| 13 | 0.7020 | 0.7331 | 0.7137 | 0.7072 | 0.7082 | 0.7029 |
| 14 | 0.6922 | 0.7162 | 0.7019 | 0.6926 | 0.6945 | 0.6892 |
| 15 | 0.6817 | 0.6983 | 0.6922 | 0.6767 | 0.6777 | 0.6797 |
| 16 | 0.6627 | 0.6802 | 0.6764 | 0.6683 | 0.6671 | 0.6650 |
| 17 | 0.6468 | 0.6706 | 0.6648 | 0.6503 | 0.6500 | 0.6492 |
| 18 | 0.6363 | 0.6545 | 0.6447 | 0.6366 | 0.6405 | 0.6313 |
| 19 | 0.6363 | 0.6545 | 0.6447 | 0.6366 | 0.6405 | 0.6313 |
| 20 | 0.6363 | 0.6545 | 0.6447 | 0.6366 | 0.6405 | 0.6313 |
| 21 | 0.6246 | 0.6429 | 0.6352 | 0.6240 | 0.6258 | 0.6166 |
| 22 | 0.6097 | 0.6302 | 0.6226 | 0.6005 | 0.6132 | 0.6007 |
| 23 | 0.5936 | 0.6187 | 0.6111 | 0.5782 | 0.5980 | 0.5847 |
| 24 | 0.5821 | 0.6026 | 0.5973 | 0.5646 | 0.5738 | 0.5648 |
| 25 | 0.5714 | 0.5888 | 0.5845 | 0.5485 | 0.5524 | 0.5434 |
| 26 | 0.5523 | 0.5622 | 0.5812 | 0.5273 | 0.5343 | 0.5276 |
| 27 | 0.5373 | 0.5506 | 0.5576 | 0.5125 | 0.5217 | 0.5108 |
| 28 | 0.5256 | 0.5306 | 0.5396 | 0.4999 | 0.5080 | 0.4960 |
| 29 | 0.5256 | 0.5306 | 0.5396 | 0.4999 | 0.5080 | 0.4960 |
| 30 | 0.5256 | 0.5306 | 0.5396 | 0.4999 | 0.5080 | 0.4960 |
| 31 | 0.5075 | 0.5095 | 0.5289 | 0.4746 | 0.4890 | 0.4780 |
| 32 | 0.4842 | 0.4927 | 0.5141 | 0.4640 | 0.4701 | 0.4671 |
| 33 | 0.4693 | 0.4790 | 0.5023 | 0.4492 | 0.4550 | 0.4500 |
| 34 | 0.4543 | 0.4664 | 0.4876 | 0.4289 | 0.4348 | 0.4319 |

**Cuadro 6.4:** Recall para un entrenamiento utilizando Alpha 0.1, genera la figura 5.5.

| Precision | | | | | | |
|---|---|---|---|---|---|---|
| **Tolerance** | **Beta 2** | **Beta 1** | **Beta 0,9** | **Beta 0,7** | **Beta 0,5** | **Beta 0,4** |
| 1 | 0.1212 | 0.1404 | 0.1420 | 0.1195 | 0.1212 | 0.1204 |
| 2 | 0.1290 | 0.1553 | 0.1582 | 0.1263 | 0.1295 | 0.1295 |
| 3 | 0.1440 | 0.1764 | 0.1815 | 0.1408 | 0.1457 | 0.1450 |
| 4 | 0.1614 | 0.2040 | 0.2102 | 0.1616 | 0.1672 | 0.1650 |
| 5 | 0.1886 | 0.2420 | 0.2494 | 0.1895 | 0.1966 | 0.1923 |
| 6 | 0.2224 | 0.2862 | 0.2944 | 0.2256 | 0.2331 | 0.2226 |
| 7 | 0.2629 | 0.3395 | 0.3524 | 0.2703 | 0.2750 | 0.2625 |
| 8 | 0.3117 | 0.4063 | 0.4160 | 0.3298 | 0.3300 | 0.3130 |
| 9 | 0.3117 | 0.4063 | 0.4160 | 0.3298 | 0.3300 | 0.3130 |
| 10 | 0.3117 | 0.4063 | 0.4160 | 0.3298 | 0.3300 | 0.3130 |
| 11 | 0.3519 | 0.4508 | 0.4638 | 0.3790 | 0.3685 | 0.3521 |
| 12 | 0.3928 | 0.4974 | 0.5150 | 0.4234 | 0.4107 | 0.3916 |
| 13 | 0.4378 | 0.5455 | 0.5588 | 0.4670 | 0.4583 | 0.4351 |
| 14 | 0.4822 | 0.5930 | 0.6029 | 0.5132 | 0.5025 | 0.4810 |
| 15 | 0.5397 | 0.6454 | 0.6495 | 0.5477 | 0.5375 | 0.5203 |
| 16 | 0.5792 | 0.6872 | 0.6838 | 0.5960 | 0.5783 | 0.5558 |
| 17 | 0.6196 | 0.7393 | 0.7152 | 0.6366 | 0.6193 | 0.5826 |
| 18 | 0.6564 | 0.7697 | 0.7433 | 0.6745 | 0.6604 | 0.6124 |
| 19 | 0.6564 | 0.7697 | 0.7433 | 0.6745 | 0.6604 | 0.6124 |
| 20 | 0.6564 | 0.7697 | 0.7433 | 0.6745 | 0.6604 | 0.6124 |
| 21 | 0.6777 | 0.8023 | 0.7580 | 0.6987 | 0.6835 | 0.6341 |
| 22 | 0.6980 | 0.8200 | 0.7770 | 0.7131 | 0.6990 | 0.6606 |
| 23 | 0.7204 | 0.8403 | 0.7875 | 0.7368 | 0.7157 | 0.6798 |
| 24 | 0.7363 | 0.8502 | 0.8050 | 0.7498 | 0.7297 | 0.6966 |
| 25 | 0.7506 | 0.8691 | 0.8177 | 0.7745 | 0.7427 | 0.7060 |
| 26 | 0.7720 | 0.8802 | 0.8370 | 0.7921 | 0.7530 | 0.7185 |
| 27 | 0.7777 | 0.9023 | 0.8432 | 0.8174 | 0.7693 | 0.7317 |
| 28 | 0.8060 | 0.9080 | 0.8469 | 0.8315 | 0.7889 | 0.7506 |
| 29 | 0.8060 | 0.9080 | 0.8469 | 0.8315 | 0.7889 | 0.7506 |
| 30 | 0.8060 | 0.9080 | 0.8469 | 0.8315 | 0.7889 | 0.7506 |
| 31 | 0.8371 | 0.9155 | 0.8592 | 0.8348 | 0.8001 | 0.7611 |
| 32 | 0.8379 | 0.9269 | 0.8653 | 0.8396 | 0.8088 | 0.7800 |
| 33 | 0.8444 | 0.9348 | 0.8767 | 0.8410 | 0.8139 | 0.7834 |
| 34 | 0.8515 | 0.9348 | 0.8795 | 0.8505 | 0.8197 | 0.7972 |

**Cuadro 6.5:** Precision para un entrenamiento utilizando Alpha 0.1, genera la figura 5.4.

| F-Measure | | | | | | |
|---|---|---|---|---|---|---|
| **Tolerance** | **Beta 2** | **Beta 1** | **Beta 0,9** | **Beta 0,7** | **Beta 0,5** | **Beta 0,4** |
| 1 | 0.2109 | 0.2387 | 0.2415 | 0.2074 | 0.2105 | 0.2091 |
| 2 | 0.2226 | 0.2599 | 0.2646 | 0.2176 | 0.2228 | 0.2227 |
| 3 | 0.2441 | 0.2888 | 0.2962 | 0.2387 | 0.2463 | 0.2451 |
| 4 | 0.2676 | 0.3243 | 0.3326 | 0.2679 | 0.2762 | 0.2727 |
| 5 | 0.3034 | 0.3702 | 0.3792 | 0.3047 | 0.3146 | 0.3082 |
| 6 | 0.3447 | 0.4190 | 0.4273 | 0.3489 | 0.3592 | 0.3448 |
| 7 | 0.3897 | 0.4725 | 0.4842 | 0.3991 | 0.4062 | 0.3892 |
| 8 | 0.4381 | 0.5313 | 0.5379 | 0.4587 | 0.4601 | 0.4400 |
| 9 | 0.4381 | 0.5313 | 0.5379 | 0.4587 | 0.4601 | 0.4400 |
| 10 | 0.4381 | 0.5313 | 0.5379 | 0.4587 | 0.4601 | 0.4400 |
| 11 | 0.4741 | 0.5651 | 0.5733 | 0.5012 | 0.4927 | 0.4738 |
| 12 | 0.5072 | 0.5959 | 0.6051 | 0.5326 | 0.5235 | 0.5059 |
| 13 | 0.5392 | 0.6255 | 0.6268 | 0.5625 | 0.5565 | 0.5375 |
| 14 | 0.5684 | 0.6488 | 0.6487 | 0.5895 | 0.5831 | 0.5666 |
| 15 | 0.6024 | 0.6708 | 0.6701 | 0.6054 | 0.5995 | 0.5894 |
| 16 | 0.6182 | 0.6837 | 0.6801 | 0.6301 | 0.6195 | 0.6055 |
| 17 | 0.6329 | 0.7033 | 0.6891 | 0.6434 | 0.6343 | 0.6141 |
| 18 | 0.6462 | 0.7074 | 0.6905 | 0.6550 | 0.6503 | 0.6217 |
| 19 | 0.6462 | 0.7074 | 0.6905 | 0.6550 | 0.6503 | 0.6217 |
| 20 | 0.6462 | 0.7074 | 0.6905 | 0.6550 | 0.6503 | 0.6217 |
| 21 | 0.6500 | 0.7138 | 0.6912 | 0.6592 | 0.6534 | 0.6252 |
| 22 | 0.6508 | 0.7127 | 0.6913 | 0.6520 | 0.6533 | 0.6292 |
| 23 | 0.6509 | 0.7127 | 0.6881 | 0.6479 | 0.6516 | 0.6287 |
| 24 | 0.6501 | 0.7053 | 0.6858 | 0.6442 | 0.6424 | 0.6238 |
| 25 | 0.6489 | 0.7020 | 0.6817 | 0.6422 | 0.6335 | 0.6141 |
| 26 | 0.6439 | 0.6861 | 0.6860 | 0.6331 | 0.6251 | 0.6084 |
| 27 | 0.6355 | 0.6839 | 0.6713 | 0.6300 | 0.6217 | 0.6016 |
| 28 | 0.6363 | 0.6698 | 0.6592 | 0.6244 | 0.6180 | 0.5973 |
| 29 | 0.6363 | 0.6698 | 0.6592 | 0.6244 | 0.6180 | 0.5973 |
| 30 | 0.6363 | 0.6698 | 0.6592 | 0.6244 | 0.6180 | 0.5973 |
| 31 | 0.6319 | 0.6547 | 0.6547 | 0.6052 | 0.6070 | 0.5872 |
| 32 | 0.6138 | 0.6434 | 0.6450 | 0.5977 | 0.5946 | 0.5843 |
| 33 | 0.6033 | 0.6335 | 0.6387 | 0.5856 | 0.5837 | 0.5716 |
| 34 | 0.5925 | 0.6223 | 0.6273 | 0.5703 | 0.5682 | 0.5603 |

**Cuadro 6.6:** F-Measure para un entrenamiento utilizando Alpha 0.1, genera la figura 5.6.

| Medidas respecto a Quality | | | | |
|---|---|---|---|---|
| **Quality** | **Precision** | **Recall** | **F-Measure** | **% documentos analizados** |
| 0 | 0.8023 | 0.6429 | 0.7138 | 100 % |
| 1 | 0.8136 | 0.6803 | 0.7410 | 92 % |
| 2 | 0.8187 | 0.7014 | 0.7555 | 78 % |
| 3 | 0.8391 | 0.7322 | 0.7820 | 64 % |
| 4 | 0.8465 | 0.7341 | 0.7863 | 54 % |
| 5 | 0.8698 | 0.7385 | 0.7988 | 45 % |
| 6 | 0.8598 | 0.7480 | 0.8000 | 40 % |
| 7 | 0.8497 | 0.7241 | 0.7819 | 35 % |
| 8 | 0.9419 | 0.7616 | 0.8422 | 31 % |
| 9 | 0.9363 | 0.7267 | 0.8183 | 28 % |
| 10 | 0.9226 | 0.7148 | 0.8055 | 25 % |
| 11 | 0.9683 | 0.7533 | 0.8474 | 23 % |
| 12 | 0.9679 | 0.7539 | 0.8476 | 21 % |
| 13 | 0.9626 | 0.7526 | 0.8447 | 19 % |
| 14 | 0.9486 | 0.7151 | 0.8155 | 17 % |
| 15 | 0.9431 | 0.7168 | 0.8145 | 16 % |
| 16 | 0.9365 | 0.7110 | 0.8083 | 14 % |
| 17 | 0.9441 | 0.7107 | 0.8109 | 12 % |
| 18 | 0.9431 | 0.7258 | 0.8203 | 11 % |
| 19 | 0.9426 | 0.7429 | 0.8309 | 10 % |

**Cuadro 6.7:** Difrentes medidas respecto a Quality, genera la figura 5.7.

| Precision | | |
|---|---|---|
| **Tolerance** | **500 Iteraciones** | **50 Iteraciones** |
| 1 | 0.1178 | 0.1404 |
| 2 | 0.1305 | 0.1553 |
| 3 | 0.1493 | 0.1764 |
| 4 | 0.1727 | 0.2040 |
| 5 | 0.2067 | 0.2420 |
| 6 | 0.2443 | 0.2862 |
| 7 | 0.2888 | 0.3395 |
| 8 | 0.3392 | 0.4063 |
| 9 | 0.3392 | 0.4063 |
| 10 | 0.3392 | 0.4063 |
| 11 | 0.3844 | 0.4508 |
| 12 | 0.4333 | 0.4974 |
| 13 | 0.4731 | 0.5455 |
| 14 | 0.5069 | 0.5930 |
| 15 | 0.5426 | 0.6454 |
| 16 | 0.5767 | 0.6872 |
| 17 | 0.6105 | 0.7393 |
| 18 | 0.6308 | 0.7697 |
| 19 | 0.6308 | 0.7697 |
| 20 | 0.6308 | 0.7697 |
| 21 | 0.6513 | 0.8023 |
| 22 | 0.6790 | 0.8200 |
| 23 | 0.6981 | 0.8403 |
| 24 | 0.7104 | 0.8502 |
| 25 | 0.7212 | 0.8691 |
| 26 | 0.7359 | 0.8802 |
| 27 | 0.7579 | 0.9023 |
| 28 | 0.7725 | 0.9080 |
| 29 | 0.7725 | 0.9080 |
| 30 | 0.7725 | 0.9080 |
| 31 | 0.7874 | 0.9155 |
| 32 | 0.8057 | 0.9269 |
| 33 | 0.8205 | 0.9348 |
| 34 | 0.8264 | 0.9348 |

**Cuadro 6.8:** Precision del modelo realizando 500 y 50 iteraciones, genera la figura 5.8.

| Recall | | |
|---|---|---|
| **Tolerance** | **500 Iteraciones** | **50 Iteraciones** |
| 1 | 0.7690 | 0.7971 |
| 2 | 0.7690 | 0.7971 |
| 3 | 0.7669 | 0.7961 |
| 4 | 0.7605 | 0.7908 |
| 5 | 0.7531 | 0.7876 |
| 6 | 0.7457 | 0.7823 |
| 7 | 0.7415 | 0.7769 |
| 8 | 0.7276 | 0.7674 |
| 9 | 0.7276 | 0.7674 |
| 10 | 0.7276 | 0.7674 |
| 11 | 0.7234 | 0.7569 |
| 12 | 0.7160 | 0.7430 |
| 13 | 0.7011 | 0.7331 |
| 14 | 0.6874 | 0.7162 |
| 15 | 0.6745 | 0.6983 |
| 16 | 0.6618 | 0.6802 |
| 17 | 0.6521 | 0.6706 |
| 18 | 0.6339 | 0.6545 |
| 19 | 0.6339 | 0.6545 |
| 20 | 0.6339 | 0.6545 |
| 21 | 0.6200 | 0.6429 |
| 22 | 0.6085 | 0.6302 |
| 23 | 0.5906 | 0.6187 |
| 24 | 0.5789 | 0.6026 |
| 25 | 0.5609 | 0.5888 |
| 26 | 0.5482 | 0.5622 |
| 27 | 0.5365 | 0.5506 |
| 28 | 0.5238 | 0.5306 |
| 29 | 0.5238 | 0.5306 |
| 30 | 0.5238 | 0.5306 |
| 31 | 0.5058 | 0.5095 |
| 32 | 0.4932 | 0.4927 |
| 33 | 0.4827 | 0.4790 |
| 34 | 0.4644 | 0.4664 |

**Cuadro 6.9:** Recall del modelo realizando 500 y 50 iteraciones, genera la figura 5.9.

| F-Measure | | |
|---|---|---|
| **Tolerance** | **500 Iteraciones** | **50 Iteraciones** |
| 1 | 0.2042 | 0.2387 |
| 2 | 0.2231 | 0.2599 |
| 3 | 0.2500 | 0.2888 |
| 4 | 0.2815 | 0.3243 |
| 5 | 0.3244 | 0.3702 |
| 6 | 0.3681 | 0.4190 |
| 7 | 0.4157 | 0.4725 |
| 8 | 0.4627 | 0.5313 |
| 9 | 0.4627 | 0.5313 |
| 10 | 0.4627 | 0.5313 |
| 11 | 0.5020 | 0.5651 |
| 12 | 0.5399 | 0.5959 |
| 13 | 0.5650 | 0.6255 |
| 14 | 0.5835 | 0.6488 |
| 15 | 0.6014 | 0.6708 |
| 16 | 0.6164 | 0.6837 |
| 17 | 0.6306 | 0.7033 |
| 18 | 0.6323 | 0.7074 |
| 19 | 0.6323 | 0.7074 |
| 20 | 0.6323 | 0.7074 |
| 21 | 0.6353 | 0.7138 |
| 22 | 0.6418 | 0.7127 |
| 23 | 0.6398 | 0.7127 |
| 24 | 0.6379 | 0.7053 |
| 25 | 0.6310 | 0.7020 |
| 26 | 0.6284 | 0.6861 |
| 27 | 0.6283 | 0.6839 |
| 28 | 0.6243 | 0.6698 |
| 29 | 0.6243 | 0.6698 |
| 30 | 0.6243 | 0.6698 |
| 31 | 0.6159 | 0.6547 |
| 32 | 0.6118 | 0.6434 |
| 33 | 0.6078 | 0.6335 |
| 34 | 0.5946 | 0.6223 |

**Cuadro 6.10:** F-Measure del modelo realizando 500 y 50 iteraciones, genera la figura 5.10.

| Precision | | |
|---|---|---|
| **Tolerance** | **Correlacionado** | **No Correlacionado** |
| 1 | 0.4931 | 0.5109 |
| 2 | 0.4931 | 0.5124 |
| 3 | 0.4931 | 0.5224 |
| 4 | 0.4931 | 0.5379 |
| 5 | 0.4945 | 0.5528 |
| 6 | 0.4945 | 0.5773 |
| 7 | 0.4958 | 0.6004 |
| 8 | 0.4986 | 0.6263 |
| 9 | 0.4986 | 0.6263 |
| 10 | 0.4986 | 0.6263 |
| 11 | 0.5014 | 0.6557 |
| 12 | 0.5056 | 0.6948 |
| 13 | 0.5101 | 0.7432 |
| 14 | 0.5194 | 0.7675 |
| 15 | 0.5224 | 0.7956 |
| 16 | 0.5241 | 0.8188 |
| 17 | 0.5309 | 0.8476 |
| 18 | 0.5410 | 0.8775 |
| 19 | 0.5410 | 0.8775 |
| 20 | 0.5410 | 0.8775 |
| 21 | 0.5541 | 0.8810 |
| 22 | 0.5527 | 0.9048 |
| 23 | 0.5558 | 0.9293 |
| 24 | 0.5634 | 0.9342 |
| 25 | 0.5733 | 0.9426 |
| 26 | 0.5879 | 0.9475 |
| 27 | 0.5995 | 0.9475 |
| 28 | 0.6160 | 0.9577 |
| 29 | 0.6160 | 0.9577 |
| 30 | 0.6160 | 0.9577 |
| 31 | 0.6265 | 0.9665 |
| 32 | 0.6298 | 0.9665 |
| 33 | 0.6408 | 0.9717 |
| 34 | 0.6403 | 0.9808 |

**Cuadro 6.11:** Precision sobre modelos correlacionados y no correlacionados, genera la figura 5.11.

| Recall | | |
|---|---|---|
| **Tolerance** | **Correlacionado** | **No Correlacionado** |
| 1 | 0.8786 | 0.9120 |
| 2 | 0.8786 | 0.9120 |
| 3 | 0.8786 | 0.9120 |
| 4 | 0.8786 | 0.9120 |
| 5 | 0.8786 | 0.9120 |
| 6 | 0.8786 | 0.9120 |
| 7 | 0.8786 | 0.9120 |
| 8 | 0.8786 | 0.9120 |
| 9 | 0.8786 | 0.9120 |
| 10 | 0.8786 | 0.9120 |
| 11 | 0.8786 | 0.9120 |
| 12 | 0.8786 | 0.9120 |
| 13 | 0.8786 | 0.9120 |
| 14 | 0.8737 | 0.9120 |
| 15 | 0.8737 | 0.9120 |
| 16 | 0.8689 | 0.9120 |
| 17 | 0.8689 | 0.9120 |
| 18 | 0.8689 | 0.9120 |
| 19 | 0.8689 | 0.9120 |
| 20 | 0.8689 | 0.9120 |
| 21 | 0.8689 | 0.9120 |
| 22 | 0.8641 | 0.9120 |
| 23 | 0.8544 | 0.9120 |
| 24 | 0.8544 | 0.9120 |
| 25 | 0.8494 | 0.9120 |
| 26 | 0.8494 | 0.9120 |
| 27 | 0.8446 | 0.9120 |
| 28 | 0.8446 | 0.9120 |
| 29 | 0.8446 | 0.9120 |
| 30 | 0.8446 | 0.9120 |
| 31 | 0.8398 | 0.9120 |
| 32 | 0.8254 | 0.9120 |
| 33 | 0.8156 | 0.9120 |
| 34 | 0.7961 | 0.9120 |

**Cuadro 6.12:** Recall sobre modelos correlacionados y no correlacionados, genera la figura 5.12.

| F-Measure | | |
|---|---|---|
| **Tolerance** | **Correlacionado** | **No Correlacionado** |
| 1 | 0.6317 | 0.6549 |
| 2 | 0.6317 | 0.6561 |
| 3 | 0.6317 | 0.6642 |
| 4 | 0.6317 | 0.6766 |
| 5 | 0.6328 | 0.6883 |
| 6 | 0.6328 | 0.7070 |
| 7 | 0.6339 | 0.7241 |
| 8 | 0.6361 | 0.7426 |
| 9 | 0.6361 | 0.7426 |
| 10 | 0.6361 | 0.7426 |
| 11 | 0.6384 | 0.7628 |
| 12 | 0.6418 | 0.7887 |
| 13 | 0.6454 | 0.8189 |
| 14 | 0.6515 | 0.8335 |
| 15 | 0.6539 | 0.8498 |
| 16 | 0.6538 | 0.8629 |
| 17 | 0.6591 | 0.8786 |
| 18 | 0.6668 | 0.8944 |
| 19 | 0.6668 | 0.8944 |
| 20 | 0.6668 | 0.8944 |
| 21 | 0.6767 | 0.8962 |
| 22 | 0.6742 | 0.9084 |
| 23 | 0.6734 | 0.9205 |
| 24 | 0.6790 | 0.9229 |
| 25 | 0.6845 | 0.9270 |
| 26 | 0.6948 | 0.9294 |
| 27 | 0.7013 | 0.9294 |
| 28 | 0.7124 | 0.9343 |
| 29 | 0.7124 | 0.9343 |
| 30 | 0.7124 | 0.9343 |
| 31 | 0.7176 | 0.9384 |
| 32 | 0.7144 | 0.9384 |
| 33 | 0.7177 | 0.9409 |
| 34 | 0.7098 | 0.9451 |

**Cuadro 6.13:** F-Measure sobre modelos correlacionados y no correlacionados, genera la figura 5.13.

| Análisis sobre FayerWayer | | | |
|---|---|---|---|
| **Tolerance** | **Precision** | **Recall** | **F-Measure** |
| 1 | 0.1020 | 0.8520 | 0.1821 |
| 2 | 0.1111 | 0.8488 | 0.1964 |
| 3 | 0.1359 | 0.8451 | 0.2342 |
| 4 | 0.1702 | 0.8386 | 0.2830 |
| 5 | 0.2067 | 0.7285 | 0.3221 |
| 6 | 0.2665 | 0.7216 | 0.3893 |
| 7 | 0.3494 | 0.7153 | 0.4694 |
| 8 | 0.4648 | 0.7121 | 0.5625 |
| 9 | 0.4648 | 0.7121 | 0.5625 |
| 10 | 0.4648 | 0.7121 | 0.5625 |
| 11 | 0.5187 | 0.6115 | 0.5613 |
| 12 | 0.5193 | 0.5292 | 0.5242 |
| 13 | 0.5856 | 0.4205 | 0.4895 |
| 14 | 0.7139 | 0.3680 | 0.4857 |
| 15 | 0.8427 | 0.3113 | 0.4546 |
| 16 | 0.8146 | 0.2434 | 0.3748 |
| 17 | 0.8720 | 0.1799 | 0.2983 |
| 18 | 0.8769 | 0.1542 | 0.2623 |
| 19 | 0.8769 | 0.1542 | 0.2623 |
| 20 | 0.8769 | 0.1542 | 0.2623 |
| 21 | 0.8875 | 0.1249 | 0.2189 |
| 22 | 0.8308 | 0.1076 | 0.1906 |
| 23 | 0.8386 | 0.1001 | 0.1789 |
| 24 | 0.8313 | 0.0817 | 0.1487 |
| 25 | 0.8571 | 0.0569 | 0.1067 |
| 26 | 0.8333 | 0.0423 | 0.0806 |
| 27 | 0.8000 | 0.0319 | 0.0613 |
| 28 | 1.0000 | 0.0319 | 0.0618 |
| 29 | 1.0000 | 0.0319 | 0.0618 |
| 30 | 1.0000 | 0.0319 | 0.0618 |
| 31 | 1.0000 | 0.0268 | 0.0523 |
| 32 | 1.0000 | 0.0144 | 0.0284 |
| 33 | 1.0000 | 0.0144 | 0.0284 |
| 34 | 1.0000 | 0.0131 | 0.0258 |

**Cuadro 6.14:** Comparativa de Precision, Recall y F-Measure sobre modelo entrenado utilizando sólo los artículos de FayerWayer, genera la figura 5.14.

| Precision | | | | |
|---|---|---|---|---|
| **Quality 0** | **Quality 1** | **Quality 2** | **Quality 3** | **Quality 4** |
| 0.10 | 0.1108 | 0.1120 | 0.1140 | 0.1257 |
| 0.11 | 0.1203 | 0.1293 | 0.1310 | 0.1599 |
| 0.14 | 0.1417 | 0.1529 | 0.1562 | 0.1614 |
| 0.17 | 0.1771 | 0.1765 | 0.1868 | 0.1994 |
| 0.21 | 0.2107 | 0.2063 | 0.2428 | 0.2393 |
| 0.27 | 0.2422 | 0.2435 | 0.3028 | 0.2994 |
| 0.35 | 0.2896 | 0.3213 | 0.4133 | 0.4417 |
| 0.46 | 0.4111 | 0.4481 | 0.5524 | 0.5852 |
| 0.46 | 0.4111 | 0.4481 | 0.5524 | 0.5852 |
| 0.46 | 0.4111 | 0.4481 | 0.5524 | 0.5852 |
| 0.52 | 0.5013 | 0.5176 | 0.6038 | 0.6222 |
| 0.52 | 0.4926 | 0.5194 | 0.6145 | 0.6792 |
| 0.59 | 0.5688 | 0.5888 | 0.7031 | 0.8000 |
| 0.71 | 0.6809 | 0.8061 | 0.8735 | 0.9429 |
| 0.84 | 0.8151 | 0.8842 | 0.9479 | 0.9643 |
| 0.81 | 0.8104 | 0.8927 | 0.9875 | 1.0000 |
| 0.87 | 0.8607 | 0.8745 | 0.9841 | 1.0000 |
| 0.88 | 0.8676 | 0.8657 | 0.9821 | 1.0000 |
| 0.88 | 0.8676 | 0.8657 | 0.9821 | 1.0000 |
| 0.88 | 0.8676 | 0.8657 | 0.9821 | 1.0000 |
| 0.89 | 0.8857 | 0.9000 | 0.9714 | 1.0000 |
| 0.83 | 0.8292 | 0.8413 | 0.9714 | 1.0000 |
| 0.84 | 0.8375 | 0.8571 | 1.0000 | 1.0000 |
| 0.83 | 0.8056 | 0.8571 | 1.0000 | 1.0000 |
| 0.86 | 0.8000 | 0.8000 | 1.0000 | 1.0000 |
| 0.83 | 0.8000 | 0.8000 | 1.0000 | 1.0000 |
| 0.80 | 0.8000 | 0.8000 | 1.0000 | 1.0000 |
| 1.00 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| 1.00 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| 1.00 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| 1.00 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| 1.00 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| 1.00 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| 1.00 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |

**Cuadro 6.15:** Análisis de Precision respecto a Quality para modelo entrenado utilizando sólo los artículos de FayerWayer, genera la figura 5.15.

| Recall | | | | |
|---|---|---|---|---|
| **Quality 0** | **Quality 1** | **Quality 2** | **Quality 3** | **Quality 4** |
| 0.85 | 0.7836 | 0.8569 | 0.8550 | 0.9188 |
| 0.85 | 0.7792 | 0.8486 | 0.8383 | 0.9188 |
| 0.85 | 0.7728 | 0.8486 | 0.8383 | 0.9188 |
| 0.84 | 0.7580 | 0.8486 | 0.8383 | 0.9188 |
| 0.73 | 0.7076 | 0.7412 | 0.8050 | 0.8688 |
| 0.72 | 0.6968 | 0.7412 | 0.8050 | 0.8688 |
| 0.72 | 0.6880 | 0.7247 | 0.8050 | 0.8688 |
| 0.71 | 0.6836 | 0.7247 | 0.8050 | 0.8688 |
| 0.71 | 0.6836 | 0.7247 | 0.8050 | 0.8688 |
| 0.71 | 0.6836 | 0.7247 | 0.8050 | 0.8688 |
| 0.61 | 0.6817 | 0.7247 | 0.8050 | 0.8688 |
| 0.53 | 0.5712 | 0.5428 | 0.6050 | 0.7438 |
| 0.42 | 0.5505 | 0.5428 | 0.6050 | 0.7438 |
| 0.37 | 0.4814 | 0.5392 | 0.5988 | 0.7313 |
| 0.31 | 0.4096 | 0.4868 | 0.5654 | 0.7313 |
| 0.24 | 0.3572 | 0.4103 | 0.4829 | 0.6563 |
| 0.18 | 0.2738 | 0.3600 | 0.3933 | 0.4813 |
| 0.15 | 0.2227 | 0.3144 | 0.3671 | 0.4500 |
| 0.15 | 0.2227 | 0.3144 | 0.3671 | 0.4500 |
| 0.15 | 0.2227 | 0.3144 | 0.3671 | 0.4500 |
| 0.12 | 0.1767 | 0.2419 | 0.2108 | 0.3125 |
| 0.11 | 0.1442 | 0.2254 | 0.2108 | 0.3125 |
| 0.10 | 0.1315 | 0.2135 | 0.2108 | 0.3125 |
| 0.08 | 0.0984 | 0.1933 | 0.1846 | 0.2813 |
| 0.06 | 0.0710 | 0.1260 | 0.0783 | 0.2063 |
| 0.04 | 0.0562 | 0.1000 | 0.0450 | 0.0813 |
| 0.03 | 0.0543 | 0.0964 | 0.0450 | 0.0813 |
| 0.03 | 0.0543 | 0.0964 | 0.0450 | 0.0813 |
| 0.03 | 0.0543 | 0.0964 | 0.0450 | 0.0813 |
| 0.03 | 0.0543 | 0.0964 | 0.0450 | 0.0813 |
| 0.03 | 0.0459 | 0.0798 | 0.0388 | 0.0688 |
| 0.01 | 0.0202 | 0.0458 | 0.0325 | 0.0563 |
| 0.01 | 0.0202 | 0.0458 | 0.0325 | 0.0563 |
| 0.01 | 0.0182 | 0.0422 | 0.0263 | 0.0438 |

**Cuadro 6.16:** Análisis de Recall respecto a Quality para modelo entrenado utilizando sólo los artículos de FayerWayer, genera la figura 5.16.

| F-Measure | | | | |
|---|---|---|---|---|
| **Quality 0** | **Quality 1** | **Quality 2** | **Quality 3** | **Quality 4** |
| 0.18 | 0.1941 | 0.1981 | 0.2012 | 0.2212 |
| 0.20 | 0.2084 | 0.2244 | 0.2266 | 0.2724 |
| 0.23 | 0.2394 | 0.2591 | 0.2633 | 0.2746 |
| 0.28 | 0.2871 | 0.2923 | 0.3055 | 0.3277 |
| 0.32 | 0.3247 | 0.3228 | 0.3731 | 0.3753 |
| 0.39 | 0.3595 | 0.3666 | 0.4401 | 0.4453 |
| 0.47 | 0.4077 | 0.4452 | 0.5462 | 0.5856 |
| 0.56 | 0.5135 | 0.5537 | 0.6552 | 0.6993 |
| 0.56 | 0.5135 | 0.5537 | 0.6552 | 0.6993 |
| 0.56 | 0.5135 | 0.5537 | 0.6552 | 0.6993 |
| 0.56 | 0.5777 | 0.6039 | 0.6900 | 0.7251 |
| 0.52 | 0.5290 | 0.5309 | 0.6097 | 0.7100 |
| 0.49 | 0.5595 | 0.5649 | 0.6504 | 0.7709 |
| 0.49 | 0.5640 | 0.6462 | 0.7105 | 0.8237 |
| 0.45 | 0.5452 | 0.6279 | 0.7083 | 0.8318 |
| 0.37 | 0.4959 | 0.5622 | 0.6486 | 0.7925 |
| 0.30 | 0.4155 | 0.5100 | 0.5620 | 0.6498 |
| 0.26 | 0.3544 | 0.4613 | 0.5344 | 0.6207 |
| 0.26 | 0.3544 | 0.4613 | 0.5344 | 0.6207 |
| 0.26 | 0.3544 | 0.4613 | 0.5344 | 0.6207 |
| 0.22 | 0.2946 | 0.3813 | 0.3465 | 0.4762 |
| 0.19 | 0.2457 | 0.3555 | 0.3465 | 0.4762 |
| 0.18 | 0.2273 | 0.3418 | 0.3482 | 0.4762 |
| 0.15 | 0.1754 | 0.3154 | 0.3116 | 0.4390 |
| 0.11 | 0.1304 | 0.2177 | 0.1453 | 0.3420 |
| 0.08 | 0.1050 | 0.1778 | 0.0861 | 0.1503 |
| 0.06 | 0.1016 | 0.1721 | 0.0861 | 0.1503 |
| 0.06 | 0.1029 | 0.1759 | 0.0861 | 0.1503 |
| 0.06 | 0.1029 | 0.1759 | 0.0861 | 0.1503 |
| 0.06 | 0.1029 | 0.1759 | 0.0861 | 0.1503 |
| 0.05 | 0.0878 | 0.1478 | 0.0746 | 0.1287 |
| 0.03 | 0.0395 | 0.0877 | 0.0630 | 0.1065 |
| 0.03 | 0.0395 | 0.0877 | 0.0630 | 0.1065 |
| 0.03 | 0.0358 | 0.0810 | 0.0512 | 0.0838 |

**Cuadro 6.17:** Análisis de F-Measure respecto a Quality para modelo entrenado utilizando sólo los artículos de FayerWayer, genera la figura 5.17.

| Análisis sobre Quality para Tolerance 15 | | | | |
|---|---|---|---|---|
| **Quality** | **Precision** | **Recall** | **F-Measure** | **% artículos evaluados** |
| 0 | 0.84 | 0.31 | 0.45 | 1.00 |
| 1 | 0.82 | 0.41 | 0.55 | 0.68 |
| 2 | 0.88 | 0.49 | 0.63 | 0.35 |
| 3 | 0.95 | 0.57 | 0.71 | 0.20 |
| 4 | 0.96 | 0.73 | 0.83 | 0.13 |

**Cuadro 6.18:** Análisis de F-Measure, Precision, Recall y el porcentaje de documentos analizados respecto a Quality para modelo entrenado utilizando sólo los artículos de FayerWayer, genera la figura 5.18.

| Cantidad de documentos | Prepare | Train |
|---|---|---|
| 1000 | 84.1 | 7.51 |
| 10000 | 389.07 | 52.86 |
| 100000 | 1330.97 | 174.02 |
| 300000 | 7731.54 | 183.14 |

**Cuadro 6.19:** Tiempos de demora del algoritmo al preparar el set de datos y al realizar el entrenamiento para 1.000, 10.000, 100.000 y 300.000 documentos, genera la figura 5.19.

| Cantidad de documentos | 10 palabras | 100 palabras | 1000 palabras |
|---|---|---|---|
| 1000 | 0.50 | 6.82 | 11.69 |
| 10000 | 0.56 | 8.28 | 13.61 |
| 100000 | 0.57 | 8.40 | 15.82 |
| 300000 | 0.57 | 8.68 | 16.32 |

**Cuadro 6.20:** Tiempos de demora del algoritmo al clasificar un texto de 10, 100 y 1000 palabras usando modelos generados con 1.000, 10.000, 100.000 y 300.000 documentos, genera la figura 5.20.