# CHATBOT DE SOLICITUDES DE INFORMACIÓN

## INFORME DE PRÁCTICAS

**Melissa Henao González**

**Título por obtener: Estadística**

**Asesor interno: Edwin de Jesús Zarrazola Rivera**

**Faculta de Ciencias Exactas y Naturales**

**Universidad de Antioquia**

**Medellín, 2020**

**Contenido**

**Resumen**

En el área de analítica de la empresa XM, a diario llegan requerimientos de los agentes que intervienen en el mercado eléctrico, estos requerimientos deben ser respondidos por los analistas, pero estas respuestas se deben dar con base a las históricamente dadas, por lo tanto, se necesita una herramienta que permita visualizar las respuestas anteriormente dadas a el ‘agente del mercado electrico’ de manera rápida para cumplir con los tiempos de ley, para esto se propone realizar un Chatbot con procesamiento de lenguaje natural, en el que el analista ingrese la pregunta al Chatbot, y éste arroje las respuestas que se le han dado a este agente en orden de importancia.

## 1. Introducción

El procesamiento del lenguaje natural(PLN) es un área de investigación en informática e inteligencia artificial (IA) relacionada con el procesamiento de lenguajes naturales como el inglés o el mandarín. Este procesamiento generalmente implica traducir el lenguaje natural en datos (números) que una computadora puede usar para aprender sobre el mundo.

Un sistema de procesamiento de lenguaje natural a menudo se conoce como una tubería porque generalmente involucra varias etapas de procesamiento donde el lenguaje natural fluye en un extremo y la salida procesada fluye por el otro.

Los lenguajes naturales son diferentes de los lenguajes de programación de computadoras. No están destinados a traducirse en un conjunto finito de operaciones matemáticas, como lo son los lenguajes de programación. Los lenguajes naturales son lo que los humanos usan para compartir información entre ellos. No usamos lenguajes de programación para contarnos sobre nuestro día o para dar instrucciones a la tienda de comestibles. Un programa de computadora escrito con un lenguaje de programación le dice a la máquina exactamente qué hacer. El procesamiento de lenguaje natural lo realizamos por medio de matemáticas y programación, en donde le enseñaremos a la maquina a tomar decisiones intentando simular el comportamiento humano.

En este caso, usaremos el PLN para la implementación del Chatbot, para que éste encuentre una respuesta al mensaje de su compañero de conversación.

**Nota**: los algoritmos del prototipo del Chatbot no aparecen ya que se prohíbe a los practicantes divulgar dicha información, pues ésta hace parte de la propiedad intelectual de XM.

## 2. Normalización del corpus

Es el proceso de transformación de los textos para la obtención de una forma canónica. Es decir, que todos los textos estén en un mismo formato. Esto incluye, eliminación de stopwords o palabras vacías, eliminación de caracteres especiales, eliminación de tildes, eliminación de palabras con una sola letra, y colocar todos los textos en minúsculas. Para esta parte se utilizó la librería Nltk de Python.

### Corpus

Un corpus es un conjunto de textos o documentos, en este caso, el corpus lo componen el conjunto de respuestas a requerimientos más cuatro variables de información que se decidieron adicionar al requerimiento para ampliar la búsqueda del analista, que están ubicados en la base de datos del Asistente de requerimientos.

### Tokenización

En el procesamiento del lenguaje natural (PNL), la tokenización es un tipo particular de segmentación de documentos. La segmentación divide el texto en fragmentos o segmentos más pequeños, con contenido de información más enfocado. La segmentación puede incluir dividir un documento en párrafos, párrafos en oraciones, oraciones en frases o frases en palabras y puntuación. Cuando segmentamos un texto en palabras y signos de puntuación a estos los llamamos tokens y a este proceso lo llamamos tokenizar

La tokenización es una herramienta poderosa. Rompe datos no estructurados, texto, en fragmentos de información que pueden contarse como elementos discretos. Estos recuentos de ocurrencias de tokens en un documento se pueden usar directamente como un vector que representa ese documento. Esto convierte inmediatamente una cadena no estructurada (documento de texto) en una estructura de datos estructurada y numérica adecuada para el aprendizaje automático. Estos recuentos pueden ser utilizados directamente por una computadora para activar acciones y respuestas útiles, o pueden ser utilizados en una tubería de aprendizaje automático como características que impulsan un comportamiento más complejo. El uso más

común para los vectores de bolsa de palabras creados de esta manera es para la recuperación o búsqueda de documentos.

Ejemplo: “Thomas Jefferson comenzó a construir Monticello a los 26 años.”, si tokenizamos la frase anterior éste sería el resultado: [‘Thomas’, ‘Jefferson’, ‘comenzó’, ‘a’, ‘construir’, ‘Monticello’, ‘a’, ‘los’, ‘26’, ‘años’, ‘.’]. Nltk proporciona una herramienta para tokenizar textos: nltk.tokenizer

**Conversión de mayúsculas en minúsculas**

Como se dijo anteriormente la idea es estandarizar el texto, para esto necesitamos que todo el texto esté escrito en minúsculas, la función lower convierte todo el texto en minúsculas.

**Eliminación de caracteres especiales**

Una tarea importante en la normalización del texto, consiste en eliminar caracteres innecesarios y especiales. Estos pueden ser símbolos especiales o incluso signos de puntuación que ocurren en las oraciones.

Este paso a menudo se realiza antes de la tokenización. La razón principal para hacerlo es porque a usualmente los signos de puntuación o caracteres especiales no tienen mucha importancia cuando analizamos el texto.

**Eliminación de palabras vacías o stopwords**

Las palabras vacías, son palabras que tienen poca o ninguna importancia, es decir, son palabras poco informativas, o que en sí no encierran un significado importante para el estudio del texto, por lo general, se eliminan del texto durante el procesamiento para retener las palabras que tienen el máximo significado y contexto. Usualmente si realizáramos el proceso de tokenización nos daríamos cuenta que son los tokens que más frecuencias tienen en el interior de un texto, algunos de estos tokens serian: la, los, sobre, de, del, por, etc. No existe una lista universal o

exhaustiva de palabras clave. Cada dominio o idioma puede tener su propio conjunto de palabras vacías, Nltk nos ofrece una lista compuesta por 313 palabras vacías o stopwords.

## 3. Colocaciones

### n-grama

Un n-grama, es una secuencia que contiene hasta n elementos que se han extraído de una secuencia de elementos, generalmente una cadena. En general, los "elementos" de un n-grama pueden ser caracteres, sílabas, palabras. En este caso vamos a considerar un n-grama a la secuencia de n palabras. Ejemplos:

- Unigrama=Arauca
- Bigrama=departamento Arauca
- Trigrama=ministerio minas energía

### Colocación

Una colocación es una secuencia o grupo de palabras que tienden a ocurrir con frecuencia, de modo que esta frecuencia tiende a ser más de lo que podría denominarse como un hecho aleatorio o casual. Se pueden formar varios tipos de colocaciones en función de las partes de discurso de los diversos términos como sustantivos, verbos, etc. Hay varias formas de extraer colocaciones, y una de las mejores es utilizar un enfoque de agrupación o segmentación de n-gramas.

La idea es tener un corpus de documentos normalizado, juntar esta lista de documentos en una sola cadena de texto (lo cual lo logramos con ' '.join(corpus)), y allí calculamos los n-gramas. Una vez calculados, contamos la frecuencia de cada n-grama, luego los clasificamos de mayor a menor en función de su frecuencia.

Para la construcción del prototipo del Chatbot la primera fase consistió en la depuración de la base de datos del Asistente de requerimientos, es decir, tanto las solicitudes como las respuestas se sometieron al proceso de normalización anteriormente descrito, y en la segunda fase se extrajeron colocaciones por medio de n-gramas de la siguiente manera:

1. Se ejecuto la función *get_top_ngrams(corpus, ngram_val=1, limit=5),* en donde corpus es la lista de respuestas a solicitudes de información,

*ngram_val* es el tamaño del n-grama, y *limit* es la cantidad de n-gramas frecuentes que queremos extraer del texto, de esta manera se extrajeron 5000 bigramas, trigramas y cuatrigramas frecuentes del corpus, a estos n-gramas los llamaremos **entidades**.

2. Con las listas obtenidas en el primer paso, se procedió a depurarlas con el objetivo de extraer las entidades.

### 3.1 Palabras claves

Con la función *get_top_ngrams(corpus, ngram_val=1, limit=5)* podemos obtener unigramas clasificados por su frecuencia en orden descendente, tal como se hizo en el caso de colocaciones, en este caso *ngram_val=1* y *limit=50000*, al colocar 50000, como tope máximo para extraer unigramas del corpus de respuestas, se observa que se generan alrededor de 15000 unigramas, este mismo procedimiento se realiza con el corpus de solicitudes y de este corpus resultan alrededor de 12000 unigramas, lo anterior genera dos conjuntos de palabras claves correspondientes al corpus de respuestas y al corpus de solicitudes, posteriormente se realiza la diferencia de conjuntos entre el conjunto del corpus de respuestas y el conjunto del corpus de solicitudes lo que da origen a las palabras claves de ambos corpus que tiene un tamaño de alrededor de 8000 palabras.

## 4. Búsqueda de documentos

El objetivo del proyecto es crear una herramienta para ayudar al analista a responder solicitudes de información y dar respuestas a las históricamente dadas, esto lo lograríamos creando un motor de búsqueda en el que el analista haga una consulta en lenguaje natural y que el resultado sean las respuestas más acordes a la pregunta dada.

En este punto ya tenemos las entidades, ahora sigue extraer las entidades de la pregunta dada por el analista, luego, encontrar estas entidades en cada documento del corpus de respuestas y retornar los documentos en donde se encuentren estas entidades.

Este procedimiento se ejecutó de la siguiente manera:

1. Identificar si la pregunta ingresada por el analista contiene fechas, lo hace por medio de expresiones regulares, si contiene fechas, las guarda en la lista *fechas*, dependiendo de la longitud de esta lista, las guardará en las listas *fecha1*, y *fecha2*, sino contiene fechas, simplemente estas variables quedarán vacías. Según la longitud de la lista fechas, se derivan cuatro casos posibles:
    - La longitud de *fechas* es igual a uno: esto significa que están preguntando por un mes o año especifico, esta fecha se guarda en la lista *fecha1*.
    - La longitud de *fechas* es igual a dos: en este caso podrían estar preguntando por: mes-año, mes-mes, año-año, en cualquier caso, la información se guardará en la lista *fecha1*.
    - La longitud de *fechas* es tres o cuatro: en este caso están preguntando por un intervalo de tiempo tal como: mes-mes-año o mes-año y mes-año, la primera fecha se guarda en la lista *fecha1*, y la segunda en *fecha2*.

2. Posteriormente se extraen las entidades y palabras clave correspondientes al mercado energético, y posteriormente las retorna. El archivo *ngramas* son las entidades sin guion, y el archivo *ngramas_sin* son las entidades con guion.

3. Con el conjunto de entidades y palabras claves obtenido en el paso anterior, se procede a realizar la intersección de este conjunto con el corpus de respuestas, la función retorna los documentos resultantes de la intersección y sus respectivos índices
4. Ya teniendo los documentos y los índices, se procede a verificar si las variables *fecha1* y *fecha2* tienen contenido, si es así, se procede a filtrar dichos documentos (variable docs) por fechas según sea el caso, si no hay fechas para filtrar, esta función retorna los mismos documentos obtenidos en el paso anterior. Esta función retorna los documentos e índices filtrados en las variables *docs_* e *indices_* respectivamente, sino hay fechas, igualmente retorna los documentos obtenidos en el paso anterior pero esta vez en las variables *docs_* e *indices_* La función está ubicada en el archivo *prototipo_Chatbot.py*, el nombre de la función es *filtrar_fechas*.
5. Por último el algoritmo identifica, si en la pregunta está en nombre de algún analista, esto lo hace comparando la lista *analistas_diom* con la pregunta, y si en efecto hay intersección, filtra los documentos e indices ( *docs_* e *indices_* respectivamente) obtenidos en el paso anterior, por el analista extraído de la pregunta, y posteriormente los retorna en las variables *_indices_* y *_docs_,* sino hay analista en la pregunta, retorna los documentos obtenidos en el paso anterior, pero en las variables *_indices_* y *_docs_.*

## 5. Documentos destacados-Matriz de adyacencia

En este punto ya hemos obtenido los documentos en donde se encuentran las entidades y palabras claves correspondientes a la búsqueda, pero este conjunto de documentos puede ser muy grande y podría causar gran dificultad al analista encontrar un documento útil para responder a una solicitud de información, por ello surgió la necesidad de clasificar los textos según su importancia.

Para clasificar los textos según su importancia se partió del hecho que las respuestas a solicitudes de información se dan de acuerdo a las anteriormente dadas, por esta razón se supone que entre estas respuestas van a estar contenidas partes de una respuesta en otras respuestas y que la respuesta que contenga partes de la mayoría de las respuestas, será la respuesta más útil para el analista, ahora la pregunta sería ¿Cómo verificar que una parte del contenido de una respuesta este contenido en otras respuestas?, la respuesta a la que se llegó es que lo lograríamos con una matriz de adyacencia.

### 5.1 Matriz de adyacencia

La matriz de adyacencia es una matriz cuadrada que se utiliza como una forma de representar relaciones binarias**.**

### 5.2 Grafo dirigido

En matemáticas y ciencias de la computación, un grafo es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto.

Un grafo dirigido es un tipo de grafo en el cual las aristas tienen un sentido definido, a diferencia del grafo no dirigido, en el cual las aristas son relaciones simétricas y no apuntan en ningún sentido.

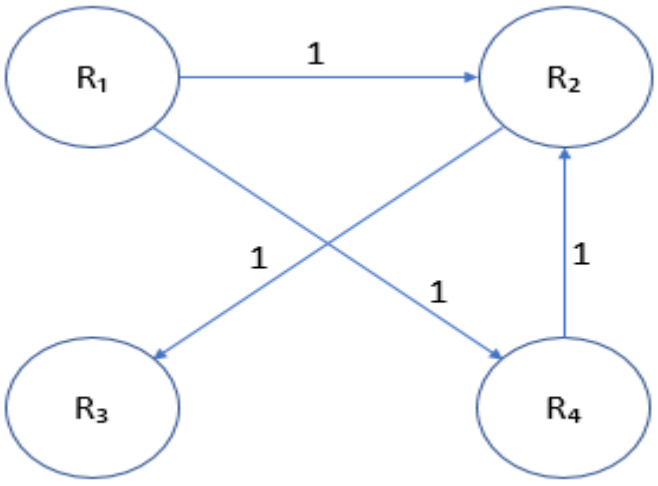
### 5.3 Aplicación

Como dijimos anteriormente el problema en este punto era asignar un nivel de importancia a cada documento, la exploración de este problema empezó por el algoritmo de PageRank de Google, y se pretendía buscar el análogo con nuestro problema de clasificación de documentos, con esto fue que surgió la idea de la

matriz de adyacencia, en el algoritmo de PageRank se usa un grafo dirigido, en el cual la dirección en la que apunta la arista significa que se hizo un enlace entre páginas y esto sucede cuando se enlaza una página a otra, para trasladar este concepto de enlace entre páginas, supusimos, si una página A tiene un enlace a otra página B, ocurre casi seguramente porque la página B tiene contenido de la página A y por esto se menciona en A, ahora en nuestro caso si una respuesta A se enlaza a una respuesta B es porque B contiene al menos el 50% de A y esto simula el proceso de enlace entre páginas, en este caso cuando nos referimos al termino “contiene” significa que la intersección entre ambos documentos es al menos el 50% del contenido de A.

Las relaciones del grafo anteriormente descrito se pueden representar por medio de una matriz, esta matriz recibe el nombre de matriz de adyacencia, ésta contiene ceros y unos, donde 1 significa que hay un enlace, y 0 la ausencia del enlace, veamos un ejemplo:

Suponga que tiene cuatro respuestas $R_1$, $R_2$, $R_3$, $R_4$ que contienen las entidades y palabras claves de su pregunta y desea puntuarlas por orden de importancia con el método anterior, en este caso usted tendría el siguiente grafo dirigido:



y la siguiente matriz de adyacencia, llamada M:

| Respuestas | $R_1$ | $R_2$ | $R_3$ | $R_4$ |
|---|---|---|---|---|
| $R_1$ | 0 | 1 | 0 | 1 |
| $R_2$ | 0 | 0 | 1 | 0 |
| $R_3$ | 0 | 0 | 0 | 0 |
| $R_4$ | 0 | 1 | 0 | 0 |

La posición $M_{13}$=1 este 1 significa:

- En el grafo vemos que la arista sale de $R_1$ con dirección hacia $R_2$, este uno representa esta relación.
- El que haya una relación de $R_1$ hacia $R_2$ implica que $R_2$ contiene por lo menos el 50% de $R_2$.

La posición $M_{43}$=0 este 0 significa:

- En el grafo vemos que no hay ninguna arista de $R_3$ con dirección a $R_2$ este cero representa que NO existe ninguna relación entre ambos nodos, es decir, no están enlazados.
- El que NO haya una relación de $R_3$ hacia $R_2$ implica que $R_2$ NO contiene por lo menos el 50% de $R_3$.

## 6. Corrector de ortografía

Como se dijo anteriormente necesitamos extraer las entidades y palabras claves de la pregunta que ingresa el analista, pero recuerde que estas entidades y palabras clave se encuentran en una lista, una vez el analista ingrese la pregunta, se realiza una intersección entre esta lista y la pregunta, esto implica, que la pregunta este bien escrita, dado que la lista no contiene errores ortográficos, por lo tanto, si la pregunta no está bien escrita se podría perder información importante para la búsqueda, por esta razón se observó la necesidad de un corrector de ortografía para maximizar la probabilidad de una búsqueda exitosa.

### 6.1 Distancia de Levenshtein

La distancia de Levenshtein, distancia de edición o distancia entre palabras, es el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra, se usa ampliamente en teoría de la información y ciencias de la computación. Se entiende por operación, bien sea una inserción, eliminación o la sustitución de un carácter. Esta distancia recibe ese nombre en honor al científico ruso Vladimir Levenshtein, quien se ocupó de esta distancia en 1965. Es útil en programas que determinan cuán similares son dos cadenas de caracteres, como es el caso de los correctores ortográficos.

Por ejemplo, la distancia de Levenshtein entre "casa" y "calle" es de 3 porque se necesitan al menos tres ediciones elementales para cambiar uno en el otro.

casa → cala (sustitución de 's' por 'l')

cala → calla (inserción de 'l' entre 'l' y 'a')

calla → calle (sustitución de 'a' por 'e')

Se le considera una generalización de la distancia de Hamming, que se usa para cadenas de la misma longitud y que solo considera como operación la sustitución. Hay otras generalizaciones de la distancia de Levenshtein, como la distancia de Damerau-Levenshtein, que consideran el intercambio de dos caracteres como una operación.

## 6.2 Distancia de Damerau-Levenshtein

En la teoría de la información y en la ciencia de computadores, se llama distancia de Damerau-Levenshtein o distancia de edición al número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra. Se entiende por operación, bien una inserción, eliminación, sustitución o transposición de dos caracteres. Lo que la distingue de la distancia de Levenshtein es que esta última cuenta como una sola operación de edición a cualquiera de las tres primeras, por el contrario, la distancia de Levenshtein contaría la transposición como dos operaciones.

## 6.3 Probabilidad condicional

La probabilidad de que ocurra el suceso A si ha ocurrido el suceso B se denomina probabilidad condicional y se define así:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

## 6.4 Construcción del corrector de ortografía

Es necesario que la pregunta este bien escrita para que la búsqueda tenga resultados exitosos, por lo tanto, se realizó una corrección ortográfica, particularmente a las palabras correspondientes al contexto energético, pues de éstas es de donde se extraen las entidades y palabras claves, por esta razón se creó un diccionario que contuviera las palabras del ámbito energético, esto se logró con la lista de unigramas que tiene una longitud de aproximadamente 6000 palabras.

Una palabra está escrita correctamente si se encuentra en el diccionario energético, en caso contrario es considerada un error. Para realizar esta revisión se procedió de la siguiente manera:

- Se revisa cada palabra de la pregunta y de allí se derivan dos casos:

a. Si la palabra hace parte de la lista de stopwords el algoritmo no la corrige

b. Si la palabra no es una stopword y además tampoco se encuentra en el diccionario energético, entonces el algoritmo procede a corregir dicha palabra.

Se procedió a corregir la palabra mal escrita de la siguiente manera:

Dada una palabra errónea ω se intenta encontrar la corrección $C_i$ de entre todas las posibles correciones que maximice la probabilidad de corregir a ω, esto es:

$$argmax\ P(c_i|w)$$

Que de acuerdo con el terorema de Bayes esto es equivalente a:

$$argmax\ \frac{P(w|c_i)P(c_i)}{P(w)}.$$

P(ω) se calcula por medio de probabilidad total, ya que esta probabilidad es la misma para toda correccion $C_i$, entonces podemos factorizar este termino y reducir lo anterior a:

$$argmax\ P(w|c_i)P(c_i)$$

Donde P($C_i$) es la probabilidad de que $C_i$ este en el corpus de respuestas, es decir, # de documentos que contienen a $C_i$ / # total de documentos en el corpus.

Ahora necesitamos hallar P(ω|$C_i$), lo cual en palabras es: ¿cual es la probabilidad de que $C_i$ sea la correccion adecuada para ω?, por ejemplo P(parmetro|parametro) es mas alta que P(parmetro|parametrizacion), entonces podemos notar que entre mas operaciones de edición tengamos que hacer, la probabilidad de que $C_i$ sea la correccion adecuada para ω se hace mas baja, para los ejemplos anteriores, el numero de operaciones que tendriamos que hacer serian 1 insercion, y 7eliminaciones+1insercion respectivamente, por esta razón para corregir una palabra erronea ω el corrector de ortografia lleva cabo dos procedimientos:

1. Dada una palabra erronea ω, se busca en el diccionario enegetico el conjunto de palabras $C_i$ i=1,2,…,n, tal que que su distacia de edicion a ω sea igual a 1.

2. Con este conjunto de palabras $C_i$ i=1,2,…,n, se halla la probabilidad $P(C_i)$= # de documentos que contienen a $C_i$ / # total de documentos en el corpus, y la que tenga probabilidad mas alta sera la correcion adecuada para ω.

## 7. Recapitulación

En esta sección se hará un breve resumen acerca del funcionamiento del Chatbot de solicitudes de información.

En primer lugar el archivo *prototipo_Chatbot.py* contiene dos funciones llamadas *Chatbot()* y *otra_consulta()*, La función *Chatbot()* no recibe ningún argumento, puesto dentro de ella hay una variable de tipo input() llamada pregunta encargada de guardar la pregunta de el usuario, posteriormente esta pregunta pasa por un proceso de normalización sin remover stopwords o palabras vacías y corrección de palabras correspondientes al mercado energético.

Dentro de la función *Chatbot()* hay otra función llamada textos(pregunta), la cual recibe como argumento la pregunta ya corregida, dentro de esta función se cumplen las principales funcionalidades del Chatbot, estas son:

1. Se remueven de la pregunta las stopwords.
2. Con la función *extraer_fechas*, se extraen las fechas de la pregunta, y si las tiene éstas son borradas de la pregunta, sino contiene fechas, las listas *fecha1* y *fecha2* quedaran vacías.
3. Se extraen las entidades de la pregunta, con la función *entidades_pregunta*
4. Con la función *docs_interseccion,* se realiza una intersección de las entidades vs las respuestas, si alguna respuesta de la base de datos requerimientos, contiene en su totalidad las entidades de la pregunta, ésta se guarda en la lista *docs* junto con su respectivo índice que es guardado en la lista *indices*.
5. Con la función *filtrar_fechas* que recibe como argumento los documentos e indices del paso anterior, ésta función verifica si existen fechas para filtrar los documentos, si existen, según sea el caso filtra los documentos que correspondan a las fechas indicadas en las pregunta y devuelve dichos documentos e indices en las variables *docs_* e *indices_* respectivamente, sino existen fechas para filtrar, los documentos e indices del paso anterior serán guardados en las variables *_docs* e *_indices* respectivamente.

6. Con la función *filtrar_analistas*, se verifica si en la pregunta está presente algún analista, esto lo hace con un archivo Excel que contiene los nombres tokenizados de los analistas, realiza la intersección de la pregunta y estos nombre tokenizados, si hay intersección, se procede a filtrar los documentos e indices del paso anterior por analista y se retornan en las variables *_docs_* e *_indices_,* si no se identifica algún analista en la pregunta, los documentos e indices del paso anterior serán guardados en las variables *_docs_* e *_indices_* respectivamente.

7. Las variables o listas *_docs_* e *_indices_* contienen los documentos e indices finales para continuar con el proceso de calificación de respuestas, por esta razón, la función *matriz_transicion* recibe como argumentos *_docs_* e *_indices_*. Esta función tiene como finalidad, puntuar las respuestas almacenadas en *_docs_* según contengan por lo menos el 50% de otras respuestas y lo hace de forma descendente. Este resultado lo almacena en la variable *suma.*

8. Por último se realiza un Data Frame con la puntuación anterior, éste contiene las siguientes columnas: 'Consecutivo', 'Responsable', 'Fecha respuesta', 'Entidad solicitante', 'Palabras claves', 'Puntuación' , los datos de las columnas anteriores, corresponden a los datos de las respuestas arrojadas por el Chatbot. Este Data Frame es exportado al disco local D, nombrado según la pregunta del usuario al Chatbot acompañado de la hora actual.

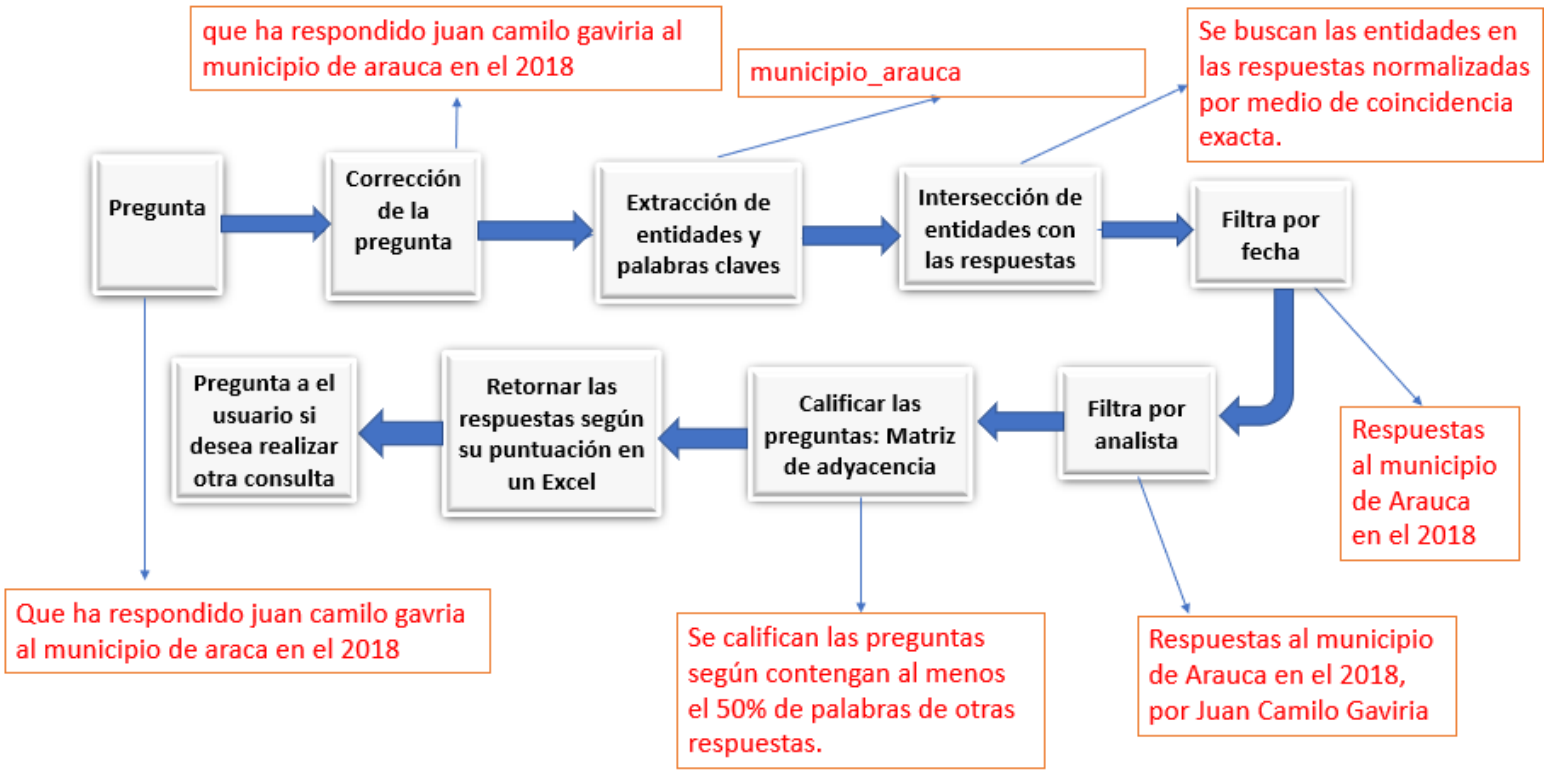
   Anteriormente se mencionó que hay una puntuación para las respuestas, esta va de 0 a 100 y se pueden presentar los siguientes casos:

   - Si en la columna de las palabras claves aparece **sin, respuesta** quiere decir que la respuesta no se encontraba en la base de datos del Asistente de requerimientos, y si adicionalmente su puntuación es 0, es porque es la única respuesta sin respuesta en la base de datos, si por el contrario, el valor es distinto de cero supongamos 50, es por que el 50% de los documentos en *_docs_* no tienen respuesta en la base de datos del Asistente.

- Si se encuentra la respuesta en la base de datos del Asistente de Requerimientos (en la columna de palabras claves no aparece **sin, respuesta**), y la puntuación es un valor diferente de 0 supongamos 70, significa que la respuesta contiene al menos el 50% del 70% del conjunto de documentos almacenados en *_docs_*
- Si se encuentra la respuesta en la base de datos del Asistente, pero su puntuación es cero, quiere decir, que esta respuesta no contiene por lo menos el 50% de alguna otra respuesta almacenada en *_docs_*

La función *otra_consulta()* es la encargada de preguntar a el usuario si desea realizar otra consulta, si la respuesta es afirmativa, se ejecutará la función *Chatbot(),* y si es negativa la función arrojará '¡Feliz Día!' y con esto se daría por terminado el proceso del *Chatbot().*

Mediante esta grafica se describe el proceso anterior:



## 8. Conclusiones

El uso de entidades y palabras claves fue muy eficiente a la hora de realizar búsqueda de documentos, y la puntuación con la matriz de adyacencia dio muy buenos resultados para explicar el contenido de la respuesta con base a lo históricamente dado.

La escritura del programa del Chatbot en Python, fue muy importante para la empresa ya que permite implementar el Chatbot en Microsoft Azure (debido a que Microsoft Azure solo brinda soporte para Python).

La empresa XM tuvo una apreciación positiva del Chatbot realizado en Python debido a que agilizó y mejoró las respuestas dadas a los agentes del mercado eléctrico.

La practica me aporto mucha experiencia en el área de programación en Python, y en general, a encontrar una solución para un problema dado con todos los conocimientos adquiridos durante la carrera.

Como trabajo a futuro se propone explorar la posibilidad de buscar las palabras claves o entidades por medio de Wor2vec (es una red neuronal, la cual, dada una palabra, retorna las palabras más acordes contextualmente a ésta),esto ayudaría a una búsqueda mas satisfactoria pues no se buscarían coincidencias exactas, sino que también se buscarían las palabras más parecidas contextualmente a las palabras claves para ampliar la búsqueda, además esta herramienta también permite comprender el contexto de una pregunta dada.  Es importante realizar una herramienta de aprendizaje automático, en donde el Chatbot aprenda de las respuestas acertadas que ha dado, para mejorar las respuestas a preguntas futuras.

## 9. Bibliografía(principal)

[1] HAPKE, Hannes Max; LANE, Hobson; HOWARD, Cole. Natural language processing in action. 2019.

[2] SARKAR, Dipanjan. Text Analytics with Python. 2016.