Método de la ingeniería - Módulo preguntas

Nombre del Sistema:

Assist Me - Módulo preguntas

Integrantes:

Alejandro Barrera

María Camila Lenis

Juan Sebastian Palma

Javier Andrés Torres

Entregado a:

Juan Manuel Reyes García

Christian Esteban Sanchez Pineda

Proyecto Integrador I

Proyecto del Curso 2019-1

Departamento TIC - Facultad de Ingeniería

Método de la ingeniería - Módulo de preguntas

Descripción del contexto

Globant S.A. es una compañía de tecnología informática dedicada al desarrollo y mantención de Software. Actualmente, cuenta con un amplio portafolio de servicios, entre los que se encuentran: consultorías, automatización de procesos, Big Data, desarrollo de aplicaciones, entre muchos más.
Debido a la gran cantidad de servicios que prestan, la compañía se divide en diferentes unidades organizacionales llamadas "*Studios*", las cuales cuentan con un correo por studio (ej: bigdata@globant.com , gaming@globant.com , cybersecurity@globant.com). Estos correos permiten que cualquier duda se comunique directamente con el *Studio*, que puede tener mayor conocimiento sobre cierto tema. Sin embargo, se ha observado que por este medio se hacen ciertas preguntas que podrían ser de interés general para muchos Globers y el conjunto de todas esas preguntas podrían ser un recurso más para el personal. Reenviar estos correos, archivar los que se consideren importantes, y buscar las personas que también puedan saber sobre el tema resulta bastante dispendioso para cada *Studio.*
Por todo lo anterior, la compañía Globant tiene la iniciativa de desarrollar una aplicación web de preguntas y respuestas para el desarrollo interno de la compañía, que cuente con dos roles, uno de usuario y otro de administrador. Esta aplicación le permitirá a los Globers antiguos y a nuevos integrantes de la organización, tener acceso a una herramienta donde pueda retroalimentar y ser retroalimentados con las preguntas de los diferentes miembros de la compañía.

Problema:

Las preguntas realizadas a los *studios* de Globant no son de fácil acceso para las personas interesadas y responder a todas ellas resulta ser una tarea dispendiosa para cada *studio.*

Especificación de Requerimientos

RF 1.1: Agregar una pregunta al listado de preguntas, debe contener su título y descripción de máximo 500 caracteres. Si el usuario lo desea, debe poder adjuntar cualquier tipo de archivo.

RF 1.1.1: Agregar hasta cinco etiquetas para la pregunta por añadir.

RF 1.1.1.1: Sugerir etiquetas relacionadas con la descripción añadida a la pregunta. Las etiquetas podrán seleccionarse para ser agregadas hasta un máximo de cinco.

RF 1.1.2: Asociar la pregunta a uno o más studios de la compañía de una lista desplegable.

RF 1.1.2.1: Sugerir studios relacionados con la descripción añadida a la pregunta. Los *studios* podrán seleccionarse y no exceder el total de cinco.

RF 1.1.3: Visualizar preguntas similares antes de terminar el proceso de envío. En caso de que la pregunta a enviar haya sido resuelta antes y el usuario cancele el envío, la pregunta no se añadirá. Si el usuario confirma que l apregunta a agregar no ha sido respondida anteriormente se comenzará el proceso de envío.

RF 1.1.4: Enviar al correo electrónico de los *studios* referenciados la pregunta a agregada. El correo debe tener un enlace a la pregunta que debe dirigirlo a la visualización de la misma dentro de la aplicación.

RF 1.2: La aplicación debe permitir la visualización de todas las preguntas añadidas en orden cronológico, es decir las últimas preguntas añadidas se mostrarán primero en forma de listado.

RF 1.3: La aplicación debe permitir la visualización de insignias para las preguntas que lo requieran, desde la lista y cuando se ha seleccionado.

RF 1.3.1: Se debe agregar la insignia “NEW” a las preguntas creadas en el día actual.

RF 1.3.2: Se debe agregar la insignia“UP”a las preguntas cuyo origen sea“Preguntar nuevamente” (Ver RF 1.5).

RF 1.3.3: Se debe agregar la insignia “ROCKSTAR” a las preguntas con más de 10 respuestas.

RF 1.3.4: Se debe agregar la insignia “SEEN BEFORE” a las preguntas que hayan sido reportadas como duplicadas.

RF 1.4: Permitir a los usuarios votar una pregunta como interesante desde la lista,o abriendo la pregunta en específico. Cada usuario podrá votar como interesante una vez por pregunta.

RF 1.5: La aplicación debe darle la opción al usuario para preguntar nuevamente una pregunta, si esta se encuentra sin responder por un lapso mayor a 24 horas.

RF 1.5.1: Si una pregunta ha sido generada por módulo de preguntar nuevamente,esta debe visualizarse en las primeras posiciones del listado.

RF 1.6: La aplicación debe mostrar al seleccionar una pregunta el número de veces que ha sido vista por un Glober. Cada Glober cuenta como una vista única.

RF 1.7: Buscar preguntas dada una descripción del usuario, las cuales aparecerán en forma de lista, ordenadas de mayor coincidencia a menor coincidencia con la descripción dada.

RF 1.7.1:La aplicación debe permitir la realización de búsquedas avanzadas en las cuales se especifiquen etiquetas, studios, Glober creador y se muestre todos los resultados como un listado ordenado por fecha de creación y cantidad de votos de pregunta interesante.

Recopilación de Información

Algoritmos más usados para hallar textos similares

Para el propósito del problema es necesario encontrar similitudes entre dos textos. Los algoritmos que realizan esta labor dependen de la longitud del texto. Para el caso en que tienen pocas palabras, se suele usar el algoritmo Tf-Idf o el BM-25 (que es una extensión del primero) para dar ponderación a cada palabra, y luego medir la similitud entre ambos textos mediante la similaridad por coseno. Por otra parte, para textos largos se suele usar el algoritmo "Locality-Sensitive Hashing"[6].

Tf-idf (Term frequency - inverse document frequency)

Es una medida numérica que expresa la relevancia de una palabra en un documento. Este valor crece proporcionalmente a la cantidad de veces que se repite la palabra en el documento pero es compensada por la cantidad de veces que aparece la palabra en toda la colección de documentos, esto con el fin de manejar el hecho de que algunas palabras son más comunes que otras.[5]

Okapi BM25

Es una función que sirve para clasificar documentos semejantes de acuerdo a su relevancia en una consulta dada. Está basado en un modelo de trabajo probabilístico.

BM25 está basado en el concepto de bolsa de palabras, con esto se representan los documentos que se desean ordenar en función de su relevancia con una consulta dada.

Por lo que tenemos que, dada una consulta *Q,* que contiene las palabras claves *q1, q2, ... , qn* el valor de relevancia para un documento D será,

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)},$$

donde $f(q_i, D)$ es la frecuencia con la que aparecen las palabras clave en el documento *D*. $|D|$ es la longitud del documento en palabras y avgdl es la longitud promedio de la colección de documentos. b y k_1 son parámetros que suelen depender de las características de la colección, aunque normalmente se le asignan valores $k_1 = 1.25$ y $b = 0.75$.

IDF(q_i) es el peso de las palabras clave que aparecen en la consulta y se calcula de la siguiente manera,

$$IDF(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$$

Donde N es el número de documentos en la colección y $n(q_i)$ es el número de documentos que contienen la palabra clave q_i .[4]

Extraer palabras clave de un texto corto

El producto de extraer palabras claves de un texto corto se puede lograr mediante el uso de machine learning. El autor del texto "Extracting Keywords From Short Text",[2] expuso tres opciones distintas como soluciones:

1. Estilo Scrabble: se toma la oración y se calcula cada valor de cada palabra como en el juego de Scrabble y se retornan las mayor puntuadas como las más importantes. Luego se toman las palabras de la siguiente oración y se compara con las anteriores para buscar aquellas palabras relacionadas o repetidas. En este caso, las palabras con mayores puntajes tienden a ser las más importantes y las que más se repiten excepto por ciertos casos; para evitar esto se implementa una lista que contenga este tipo de palabras consideradas como excepciones y no contarlas en el ranking. Al finalizar, el ranking debe de tener una lista de puntajes en el cual las palabras más importantes estarían de primeras.
2. Método RAKE (Rapid Automated Keyword Extraction): Este método consiste en eliminar las palabras más comunes en el idioma como conectores entre otros para al final dejar las palabras que se podrían considerar no comunes o las importantes del texto. En este caso no se garantiza perfectamente que sean las palabras más importantes del texto corto. Por último, RAKE se encuentra implementado en varios lenguajes de programación lo cual lo hace bastante simple y sencillo de usar.
3. Inventar un nuevo Lenguaje: En este caso sería crear un lenguaje mediante el uso de palabras importantes como secuencia y hacer que la máquina aprenda a reconocerlas mediante una base de datos, tomándolas y creando un nuevo lenguaje basándose en ellas. Primero es necesario generar un set de datos, en el cual se pueda relacionar las palabras clave a algún tema y así empezar a categorizarlas. Luego, se cargan los datos del programa en la red neuronal con una estructura base. Para continuar, se ejecuta la red neuronal y se deja entrenando la secuencia de secuencias hasta que termine, al terminar la red neuronal ya debería de tener la capacidad de identificar algunas o varias palabras claves dentro del texto que se ingrese. Para mejores resultados es preferible que se entrene la red neuronal con una cantidad de datos mayor.
4. De acuerdo a la página *Elastic*,[3] en el artículo titulado "More Like this Query" o Consulta Mas Como Esto, este tipo de consultas consisten de encontrar documentos parecidos en base a un conjunto de documentos mediante la búsqueda de términos similares dentro de los documentos. Para lograr este proceso, se seleccionan una cantidad representativa de términos en el documento ingresado, y luego se genera una petición de búsqueda en la base de datos para encontrar textos que contengan términos similares. Para que esta consulta funcione, primero se debe de clasificar el texto ingresado dentro de un tipo de consulta, en este caso se utiliza como método la fórmula de puntaje de Lucene (buscar similitudes dentro del texto, asignarles un puntaje y comparar donde pertenece). Luego, se

toma el texto del documento ingresado y se analiza para poder extraer n cantidad de términos clave y utilizarlos en la consulta para la búsqueda de textos similares. Al finalizar el proceso, se retorna un conjunto de documentos similares al que se ingresó.

5. Keywords Extraction API Documentation de Rapid Api: Esta API permite tomar palabras clave de uno o varios URL o textos. Como parámetros de entrada pide un string de JSON especificando la entrada como tal de textos (el texto plano o el url), seguido del tipo de entrada y el número de palabras clave a encontrar. Cabe resaltar que para esta API la versión gratuita admite solo hasta 100 requests, y es necesario registrarse con tarjeta de crédito.[1]

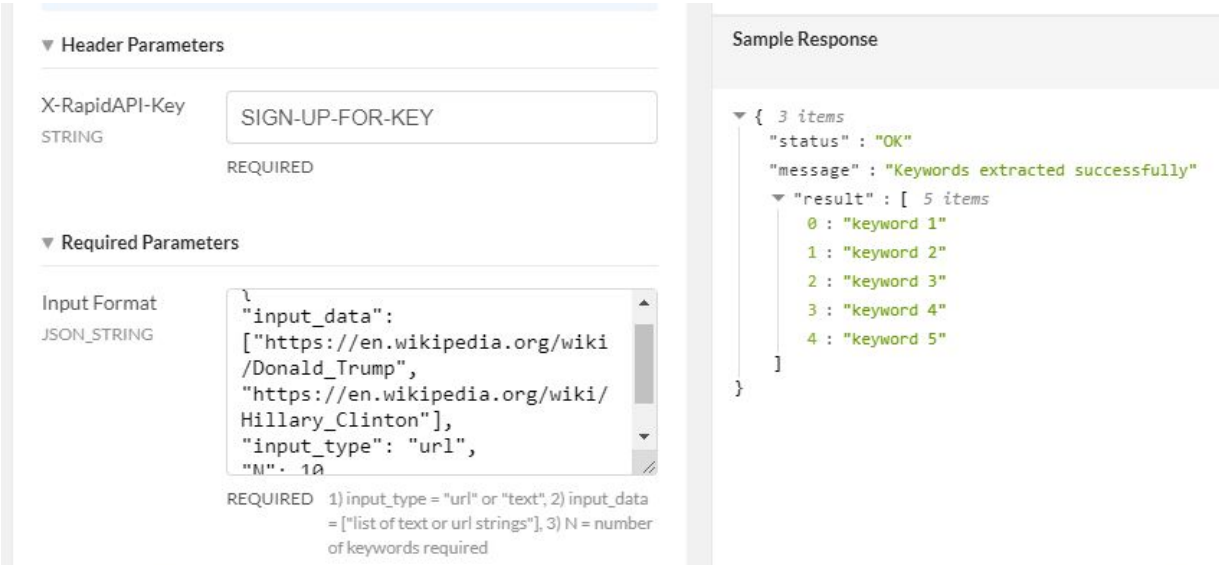

Fig 1. Ejemplo de API de extracción de palabras claves[1]

<u>Búsqueda de soluciones creativas</u>

En la parte del módulo de preguntas se desarrollará una búsqueda de soluciones a la funcionalidad más relevante y general a todos los requerimientos que es la predicción o sugerencia de etiquetas y *studios* relacionados con la descripción añadida a la pregunta, de la mano con la funcionalidad de encontrar preguntas relacionadas con base a la misma descripción. De esta forma se engloba la problemática a encontrar la manera de sugerir etiquetas, *studios* o preguntas relacionadas de acuerdo a una descripción dada. Para lograrlo se realizó una *lluvia de ideas* de la cual se extraen las siguientes:

1. Implementar una función que tome palabras aleatorias de la descripción escrita, y buscar preguntas con la mayor cantidad de coincidencias
2. Implementar un conjunto de diccionarios que almacenen el número de veces que se han encontrado palabras dentro de la descripción, y con base a eso buscar en los diccionarios de otras preguntas una cantidad de palabras similares que sea cercana.
3. Implementar un arreglo por *studio* en el cual se almacenen palabras clave que tengan una relación directa con el *studio* y al escribir o enviar la pregunta, el programa buscará en los arreglos de los *studios*

4. Crear un método que ayude a clasificar las preguntas en base a su contenido mediante el análisis de la descripción para proveer la mejor sugerencia
5. Comparar el texto de la descripción con el de otras preguntas y en base a un porcentaje de similitud en palabras se sugiera.
6. Buscar palabras que tengan una importancia dentro de una frase que permita comparar u/o ubicar relaciones con mayor facilidad e implementar sugerencias precisas
7. Implementar un modelo de text analytics que por medio de aprendizaje no supervisado encuentre patrones en las preguntas para buscar relaciones.
8. Implementar el algoritmo BM25 para buscar similitudes por medio de un análisis probabilístico.
9. Utilizar una API que tome parámetros JSON de textos y retorne las palabras claves de dicho texto.

Transición de ideas a diseños preliminares

En primer lugar, se ha decidido descartar las ideas 1, 4, 3, 6 y 7 por motivos descritos a continuación:

- La idea 1 toma palabras aleatorias de un texto dado y luego busca coincidencias de la misma en otro conjunto de datos. Esto no permite tener un acercamiento de lo que realmente dice en el texto dado, o incluso se pueden llegar a escoger aleatoriamente palabras muy generales, usadas en muchos textos y la sugerencia sería ineficiente.
- Las ideas 3 y 6 tienen algo en común y es que se enfocan en una búsqueda en específico, es decir, son soluciones a problemas de búsqueda en particular como lo es texto-texto o texto-studio y lo que se necesita es que sea una solución genérica aplicable a los todos los casos. Por otro lado, un arreglo con palabras clave del *studio* no alcanza a ser lo suficientemente relevante para tomar una decisión sobre si existe una relación, debido a la cantidad de palabras que pueden pertenecer a varios *studios* y además en un inició se debería ingresar esas palabras manualmente.
- La idea 4 sugiere la elaboración de un método para clasificar las preguntas, por lo tanto luego de que estén clasificadas las preguntas que estén dentro de un mismo conjunto son las relacionadas. Es descartada porque no expone un método para definir un criterio de clasificación, y ese mismo puede ser cambiante, es decir, dependiendo de este criterio las preguntas pueden ubicarse en diferentes conjuntos, lo que la un análisis subjetivo y lo que se busca es que se encuentren relaciones objetivas en base a una descripción dada.
- La ideas 7 está relacionada al uso de text analytics para encontrar relaciones. Aunque es una metodología muy buena, no partimos de un conjunto de datos grande para poderla implementar; además de los pocos conocimientos en esa ciencia, por lo tanto, sería irresponsable tomarla como elección.

Después de realizar un primer filtro a la lluvia de ideas, se explicará más a fondo en qué consisten las otras 6 ideas restantes y se le asignará un nombre a cada una para que sea más sencilla su identificación:

- **Idea 2, Implementar diccionarios de Comparación:** Para cada pregunta se debe asignar un diccionario que tenga como llave una palabra de la descripción y como valor la cantidad de veces que fue usada en la pregunta. Para comparar con otra pregunta se haría sobre esos valores y la pregunta que más puntaje tenga es la que está más relacionada.

- **Idea 5, implementar comparación de textos mediante porcentajes:** Se comparan los textos de la descripción de cada pregunta, encontrando así un porcentaje de similitud y los que tengan mayor porcentaje de similitud serán mostrados como relacionados.
- **Idea 8, implementar el algoritmo BM25:** Implementar el algoritmo BM25 para hacer una consulta sobre documentos utilizando un análisis probabilístico, descrito en la fase recopilación de información previamente en este informe.
- **Idea 9, utilizar el API de extracción de palabras:** Utilizar la api "*Keywords Extraction*" de Rapid Api que permite tomar palabras clave de uno o varios URL o textos. Como parámetros de entrada pide un string de JSON especificando la entrada como tal de textos (el texto plano o el url), seguido del tipo de entrada y el número de palabras clave a encontrar.

Evaluación y selección de ideas

Criterio A: Tipo de entrada. Admite no solo un string sino otros tipos de texto o de fuentes para realizar la búsqueda de similitudes.

- [3] Admite texto plano y url
- [2] Admite documentos, es decir, no sólo una descripción, sino un título y otros atributos de la pregunta.
- [1] Admite solo un texto.

Criterio B: Método de Análisis. Utiliza un método de análisis matemático para extraer conclusiones sobre las relaciones

- [3] Utiliza un modelo matemático
- [1] Compara texto a texto

Criterio C: Escalabilidad. La solución es fácilmente adaptable a los demás casos, es decir busca relaciones texto-texto, texto-*studio*, texto-etiquetas.

- [3] Puede adaptarse fácilmente a todos los casos
- [2] Es aplicable para dos de los casos
- [1] Es aplicable para un caso.

Criterio D: Costo. La solución no es paga, o si es paga, cubre todas las funcionalidades requeridas.

- [3] Es gratuita
- [2] Es paga, pero cumple con todas las funcionalidades
- [1] Es paga

	Criterio A	Criterio B	Criterio C	Criterio D	Total
Comparar diccionarios	1	3	2	3	9

Comparar textos	1	1	1	3	6
BM25	2	3	3	3	11
API	3	3	2	1	9

La idea seleccionada es implementar el algoritmo BM25 para buscar relaciones entre documento mediante un análisis probabilístico. Esta solución es efectiva porque al encontrar las preguntas relacionadas puede escalarse para encontrar etiquetas y *studios* relacionados; lo haría buscando sobre las preguntas relacionadas cuáles son las etiquetas usadas y los *studios* asignados y así realizar sugerencias. Es una solución gratuita, basada en un método matemático, por lo tanto arroja resultados objetivos sobre las relaciones y no necesita de un conjunto grande de datos para funcionar.

Síntesis reflexiva

Tras la realización del método de la ingeniería se encontró la mejor solución para todas las necesidades planteadas. Las funcionalidades del módulo de preguntas recaen en la necesidad de hallar la manera de encontrar relaciones entre un texto dado y todas las preguntas existentes, por lo tanto la búsqueda, transición y selección se hizo con base a ello. Durante el proceso de recopilación de información se encontraron varias alternativas, unas puramente matemáticas y otras ya existentes en el mercado que sirvieron para tomar diferentes enfoques a una misma problemática. De esta manera forma, se puede ver como en el proceso de transición de ideas se priorizó aquellas cuya metodología era puramente analítica y qué tanto la respuesta arrojada puede escalarse al resto de necesidades existentes. Finalmente, se considera que hubo un buen manejo del proceso de selección de la mejor solución porque las alternativas elegidas para cada problema cumplieron por completo con los requerimientos y de forma efectiva con los recursos disponibles.

Bibliografia

[1]“Extracting Keywords From Short Text,”*towardsdatascience.com*. Publicado el 27 de Julio, 2017.[Online]. Disponible en: https://towardsdatascience.com/extracting-keywords-from-short-text-fce39157166b [Visitado en Marzo 10,2019]

[2]“Keywords Extraction”*rapidapi.com*, última actualización: 19 de Febrero, 2019. [Online]. Disponible en: https://rapidapi.com/UnFound/api/keywords-extraction2 [Visitado en Marzo 10,2019]

[3]“More Like This Query,” *elastic.co,* 2019. [Online]. Disponible en: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-mlt-query.html [Visitado en Marzo 10, 2019]

[4]“Okapi BM25”, *wikipedia.com.* Publicado el 22 de Junio, 2018.[Online]. Disponible en: https://es.wikipedia.org/wiki/Okapi_BM25 [Visitado en Marzo 10, 2018]

[5]“Tf-idf”,*wikipedia .com*. Publicado el 17 Septiembre, 2018.[Online]. Disponible en: https://es.wikipedia.org/wiki/Tf-idf[Visitado en Marzo 10, 2018]

[6]“What are the most popular text similarity algorithms” *www.quora .com.* Publicado el 10 de Diciembre, 2015.[Online]. Disponible en: https://www.quora.com/What-are-the-most-popular-text-similarity-algorithms [Visitado en Marzo 10, 2019]