RESUMEN

Versiones para generar preguntas y respuestas

Importante Saber

Similitud Coseno:

Similitud de coseno, consiste en evaluar la similitud de dos vectores calculando el coseno del ángulo entre ellos. La similitud de coseno dibuja vectores en el espacio vectorial de acuerdo con valores de coordenadas, como el espacio bidimensional más común.

La similitud (por ejemplo, entre dos o más textos y oraciones) se determina comparando vectores de palabras o "incrustaciones de palabras", representaciones de significado multidimensional de una palabra. Los vectores de palabras se pueden generar usando un algoritmo como word2vec que usualmente se verían así:

```
array([2.02280000e-01,  -7.66180009e-02,   3.70319992e-01,
       3.28450017e-02,  -4.19569999e-01,   7.20689967e-02,
      -3.74760002e-01,   5.74599989e-02,  -1.24009997e-02,
       5.29489994e-01,  -5.23800015e-01,  -1.97710007e-01,])
```

Word2vec representa cada palabra distinta con una lista particular de números llamada vector. Los vectores están escogidos cuidadosamente de forma que una función matemática sencilla (la similitud coseno entre los vectores) indica el nivel de la similitud semántica entre las palabras representada por dichos vectores.

https://spacy.io/usage/linguistic-features#vectors-similarity

https://es.wikipedia.org/wiki/Word2vec

Entidad nombrada:

Entidad nombrada es un objeto del mundo real, como una persona, ubicación, organización, producto, etc., que se puede denotar con un nombre propio.

Token en textos:

Los tokens son unidades pequeñas significativas formadas al dividir un texto en partes más pequeñas (Tokenización). Ejemplos de tokens pueden ser palabras, caracteres, números, símbolos o n-gramas. Y de esos tokens sacar sus propiedades gramaticales (verbos, fomas verbales, sustantivos, sujeto, etc)

Decir que la similitud coseno en algunas circunstancias no es exacta del todo (puede deberse a la forma o el funcionamiento que se implementó a la hora de hallarla, como se construyen los vectores y la información a procesar que contengan los mismos), algunos métodos que permiten calcularla pueden dar mejores resultados que otros, esto en algunas ocasiones puede variar.

Primera versión

Elementos usados:

- Spacy (modelo español LG (el más completo))
- Modelo pre-entrenado para generar respuestas. Enlace: https://huggingface.co/mrm8488/electricidad-small-finetuned-squadv1-es
  Otro modelo para generar respuestas, esta pesa más: https://huggingface.co/IIC/roberta-base-spanish-sqac
- Modelo pre-entrenado para generar preguntas. Enlace: https://huggingface.co/mrm8488/bert2bert-spanish-question-generation
- Código base original de tercero. Enlace: https://medium.com/featurepreneur/question-generator-d21265c0648f

Texto de ejemplo a procesar (Ejemplo de todos)

Código de las Familias es una norma sustantiva del Derecho de Familia en Cuba; Cuerpo legal que regula todas las instituciones relativas a la familia: el matrimonio, el divorcio, las relaciones paterno filiales, la obligación de dar alimentos, la adopción y la tutela. Se promulga en 1975, modificado y propuesto a referendo popular el 25 de septiembre de 2022, siendo ratificado por el pueblo cubano con el 66.87 % de los votos.

Las normas contenidas en este Código se aplican a todas las familias cualquiera que sea la forma de organización que adopten y a las relaciones jurídico-familiares que de ellas se deriven entre sus miembros, y de estos con la sociedad y el Estado y se rigen por los principios, valores y reglas contenidos en la Constitución de la República de Cuba, los tratados internacionales en vigor para el país que tienen incidencia en materia familiar y los previstos en este Código.

Es un Código inclusivo, revolucionario y novedoso en su texto como en su proceso de elaboración. Protege a niños, niñas y adolescentes, les reconoce derechos a las personas adultas mayores y en situación de discapacidad, visibiliza y reconoce derechos a sectores vulnerables, condena la violencia familiar y establece herramientas para los que han sido víctima de ella, condena la discriminación contra la mujer, democratiza las relaciones familiares, le otorga efectos jurídicos al afecto, y reconoce en su articulado la diversidad de realidades que existe entre las familias cubanas.

1. Cargamos los modelos.

```
nlp = spacy.load("es_core_news_lg")

qa_pipeline = pipeline(
        "question-answering",
        model="small-gen-answers-generation",
        tokenizer="small-gen-answers-generation"
    )

mdl = AutoModelForSeq2SeqLM.from_pretrained(
            'bert-base-spanish-wwm-cased')
tknizer = AutoTokenizer.from_pretrained(
            'bert-base-spanish-wwm-cased')
```

2. Se pone el texto en el procesador de documentos de Spacy

3. Extraemos las oraciones del texto y guardándolas en una lista

```
doc = nlp(texto)

sentences = list(doc.sents)
```

4. Hacemos un ciclo para analizar cada oración
   Cada oración será tokenizada mediante el modelo para generar preguntas, ya que esta entrenado para realizar este tipo de acción también.

   Este método recibe la oración como el contexto o situación desde donde se va a generar la pregunta

   Se procede a tokenizar

```
text = "context: {}".format(sentence)
        max_len = 256
        encoding = tknizer.encode_plus(
           text, max_length=max_len, pad_to_max_length=False, truncation=True,
return_tensors="pt")
```

5. Los resultados serán enviados como parámetros al modelo que generará la pregunta.

```
        outs = mdl.generate(input_ids=input_ids,
                            attention_mask=attention_mask,
                            early_stopping=True,
                            num_beams=5,
                            num_return_sequences=1,
                            no_repeat_ngram_size=2,
                            max_length=300)
```

6. Los resultados del modelo serán analizados y con un ciclo vamos a sacar cada pregunta generada.

- La salida del modelo sería “question: Pregunta generada”, la frase “question:” se reemplaza o elimina dejando solo la pregunta.

```
dec = [tknizer.decode(ids, skip_special_tokens=True)
               for ids in outs]

Question = dec[0].replace("question:", "")
       Question = Question.strip()
       print(Question)
```

7. Luego le pasamos al modelo de generar respuestas las preguntas y el contexto (la oración)

```
result = qa_pipeline({
                    'context': str(sent),
                    'question': Question})
Print(result)
```

SALIDA

¿Cuál es el nombre oficial del Código de las Familias?
{'score': 0.15141291916370392, 'start': 0, 'end': 22, 'answer': 'Código de las Familias'}

¿Qué votó el pueblo cubano en contra de la enmienda?
{'score': 0.3791300356388092, 'start': 139, 'end': 146, 'answer': '66.87 %'}

El Código de la República de Cuba está sujeto a ¿qué tipo de tratado?
{'score': 0.399568110704422, 'start': 354, 'end': 378, 'answer': 'tratados internacionales'}

¿Cuál es la definición de un código inclusivo en su conjunto?
{'score': 0.2582134008407593, 'start': 24, 'end': 49, 'answer': 'revolucionario y novedoso'}

¿Qué hace la visibilización de la gente en Cuba?
{'score': 0.3266850709915161, 'start': 137, 'end': 177, 'answer': 'reconoce derechos a sectores vulnerables'}

- En este ejemplo las preguntas generadas algunas no tienen todo el sentido si se mira el texto, por ejemplo, la segunda pregunta (¿Qué votó el pueblo cubano en contra de la enmienda? Cuando en realidad es a favor). **Con otros textos tiene mejores resultados, aunque no del todo, puede variar, como puede pasar con las demás versiones.**

Segunda versión

Elementos usados:

- Spacy (modelo español LG (el más completo))
- Modelo pre-entrenado para generar respuestas. Enlace: https://huggingface.co/mrm8488/electricidad-small-finetuned-squadv1-es
  Otro modelo para generar respuestas, esta pesa más: https://huggingface.co/IIC/roberta-base-spanish-sqac
- Modelo pre-entrenado para analizar similitud oraciones. Enlace: https://huggingface.co/eduardofv/stsb-m-mt-es-distilbert-base-uncased
- Código base original de terceros. Enlace: https://towardsdatascience.com/semantic-similarity-using-transformers-8f3cb5bf66d6

Texto de ejemplo a procesar (Ejemplo de todos)

Código de las Familias es una norma sustantiva del Derecho de Familia en Cuba; Cuerpo legal que regula todas las instituciones relativas a la familia: el matrimonio, el divorcio, las relaciones paterno filiales, la obligación de dar alimentos, la adopción y la tutela. Se promulga en 1975, modificado y propuesto a referendo popular el 25 de septiembre de 2022, siendo ratificado por el pueblo cubano con el 66.87 % de los votos.

Las normas contenidas en este Código se aplican a todas las familias cualquiera que sea la forma de organización que adopten y a las relaciones jurídico-familiares que de ellas se deriven entre sus miembros, y de estos con la sociedad y el Estado y se rigen por los principios, valores y reglas contenidos en la Constitución de la República de Cuba, los tratados internacionales en vigor para el país que tienen incidencia en materia familiar y los previstos en este Código.

Es un Código inclusivo, revolucionario y novedoso en su texto como en su proceso de elaboración. Protege a niños, niñas y adolescentes, les reconoce derechos a las personas adultas mayores y en situación de discapacidad, visibiliza y reconoce derechos a sectores vulnerables, condena la violencia familiar y establece herramientas para los que han sido víctima de ella, condena la discriminación contra la mujer, democratiza las relaciones familiares, le otorga efectos jurídicos al afecto, y reconoce en su articulado la diversidad de realidades que existe entre las familias cubanas.

1. Cargamos los modelos.

```
nlp = spacy.load("es_core_news_lg")

model = SentenceTransformer('similarity-es-evaluation')

qa_pipeline = pipeline(
        "question-answering",
        model="small-gen-answers-generation",
       tokenizer="small-gen-answers-generation")
```

2. Se pone el texto en el procesador de documentos de Spacy

```
doc = nlp(texto)
```

3. Extraemos las oraciones del texto y guardándolas en una lista

```
sentences = list(doc.sents)
```

4. Hacemos un ciclo para analizar cada oración
   Cada oración será tokenizada mediante Spacy, además de extraer las entidades nombradas con el mismo.

   Se procede a tokenizar

```
# Extrayendo verbos, formas verbales al tokenizar la oracion
        for tok in docSent:
            # print(tok.text, tok.tag_, tok.dep_, tok.pos_)
            if(tok.tag_ == "VERB" and tok.dep_ == "ROOT"):
                formaverbal = tok.text
            elif(tok.tag_ == "AUX"):
                formaVerbalAUX = tok.text
           elif(tok.tag_ == "VERB" and tok.dep_ == "acl" or tok.dep_ == "advcl"
or tok.dep_ == "relcl"):
                verbo = tok.text
```

5. Extraemos las entidades y generamos las preguntas

   Por ejemplo, entidad “PER” para las personas, “ORG” para organizaciones, “LOC” para localidades, “MISC” (Misceláneas) es para varios tipos como eventos, nacionalidades, productos, etc

   Para generar la pregunta por ejemplo si es una persona:

   ¿Quién es __? el espacio que queda después de “es” lo sustituimos por la persona extraída

   Si es para una localidad: ¿Dónde queda __? añadimos la entidad localidad extraida en el espacio después de “queda”

En el paso de arriba tokenizamos para extraer verbos o formas verbales, estás se usan para generar preguntas que lleven la palabra con esa categoría gramatical.

Si se habla de una persona y hay una forma verbal casi siempre significa la acción que realizó dicha persona por lo que una pregunta sería en caso de ejemplo:

Pablo participó en un evento para desarrolladores web.

¿Quién participó? Sería la pregunta con la forma verbal extraída.

```
if(ent.label_ == "MISC"):
                    questions.append('Qué es {}'.format(ent.text))

            elif(ent.label_ == "PER"):
                personaje = ent.text
                questions.append('Quién es {}'.format(ent.text))

            elif(ent.label_ == "ORG"):
                organizacion = ent.text
                questions.append('Qué paso en o con {}'.format(ent.text))

                 if(formaverbal != "" and organizacion != "" and personaje !=
"");
                    questions.append('Quién {}'.format(
                        formaverbal) + " en {}".format(organizacion))
                # questions.append('A quien {}'.format(formaverbal))

            if(ent.label_ == "LOC"):
                # questions.append('Que paso en {}'.format(ent.text))
                questions.append('Qué es {}'.format(ent.text))
                questions.append(
                    'Dónde queda o se localiza {}'.format(ent.text))
```

6. Analizamos la similitud de las preguntas generadas con las oraciones de donde provienen.
   La oración que tenga mayor similitud con su pregunta es la que será tomada en cuenta para dar resultados.

7. Pasamos al modelo como parámetro la lista de oraciones y la pregunta. El modelo codifica estos parámetros para luego procesarlos.

```
        corpus_embeddings = model.encode(sentencesSTR,convert_to_tensor=True)

        embedding1 = model.encode(pregunta, convert_to_tensor=True)
```

8. Mediante un ciclo analizamos cada pregunta y calculamos la similitud con la lista de oraciones que tenemos mediante una función (***util.pytorch_cos_sim()***) de la biblioteca **sentence_transformers** a la que le cargamos el modelo para analizar la similitud, recibe la pregunta y la lista de oraciones en el formato que maneja el modelo que analiza la similitud la cual obtuvimos en el paso 7.

```
from sentence_transformers import SentenceTransformer, util
    for pregunta in questions:

        cos_scores = util.pytorch_cos_sim(
            embedding1, corpus_embeddings)[0]
```

9. Se hace un ranking de las oraciones por puntaje de similitud y la que tenga la puntuación más alta es la que tomamos. Se ordena de forma decreciente.

```
        top_results = np.argpartition(-cos_scores, range(top_k))[0:top_k]

        for idx in top_results[0:top_k]:
            print(sentencesSTR[idx], "(Score: %.4f)" % (cos_scores[idx]))
```

10. Una vez que tenemos la preguntas y la oración más similar a estas, procedemos a pasarlas al modelo que genera la respuesta a dicha pregunta.

```
        result = qa_pipeline({
                'context': sentencesSTR[idx],
                'question': pregunta})
        questionsResult.append(pregunta)
        answersResult.append(result['answer'])

print("Preguntas: " + str(questionsResult) + "\n" +
      "\n" + "Respuestas: " + str(answersResult))
```

SALIDA
La posición de cada pregunta corresponde a la de la respuesta, o sea la primera pregunta con la primera respuesta.

Preguntas: [
'Dónde queda o se localiza Estado', '
Qué es Derecho de Familia',
'Qué es Código',
'Qué es Constitución de la República de Cuba',
'Qué es Código de las Familias',
'Dónde queda o se localiza Cuba',
'Qué es Cuba',
'Qué es Estado',
'Qué paso en o con Cuerpo']

Respuestas: [
'la sociedad y el Estado',
'Código de las Familias',
'proceso de elaboración',
'Código de las Familias',
'una norma sustantiva del Derecho de Familia',
'con la sociedad y el Estado',
'ratificado por el pueblo cubano',
'la sociedad y el Estado',
'la sociedad y el Estado']

- No hay mucho sentido en algunas preguntas en correspondencia con la respuesta, incluso algunas preguntas generadas no tienen respuesta en el texto (En este método las preguntas son muy específicas o cerradas a las entidades por lo que en ocasiones se alejan del contenido del texto y no haber una respuesta a ellas, quizá si es posible habría que buscar mejores formas de preguntas según la entidad) por lo que es entendible que no haya mucha correspondencia. **Con otros textos con una redacción que favorezca más a este método tiene mejores resultados, aunque no del todo, puede variar, como puede pasar con las demás versiones.**

Tercera versión

Elementos usados:

- Spacy (modelo español LG (el más completo)) y también se usó su función para calcular la similitud coseno
- Modelo pre-entrenado para generar respuestas. Enlace: https://huggingface.co/mrm8488/electricidad-small-finetuned-squadv1-es
  Otro modelo para generar respuestas, esta pesa más: https://huggingface.co/IIC/roberta-base-spanish-sqac

Texto de ejemplo a procesar (Ejemplo de todos)

Código de las Familias es una norma sustantiva del Derecho de Familia en Cuba; Cuerpo legal que regula todas las instituciones relativas a la familia: el matrimonio, el divorcio, las relaciones paterno filiales, la obligación de dar alimentos, la adopción y la tutela. Se promulga en 1975, modificado y propuesto a referendo popular el 25 de septiembre de 2022, siendo ratificado por el pueblo cubano con el 66.87 % de los votos.

Las normas contenidas en este Código se aplican a todas las familias cualquiera que sea la forma de organización que adopten y a las relaciones jurídico-familiares que de ellas se deriven entre sus miembros, y de estos con la sociedad y el Estado y se rigen por los principios, valores y reglas contenidos en la Constitución de la República de Cuba, los tratados internacionales en vigor para el país que tienen incidencia en materia familiar y los previstos en este Código.

Es un Código inclusivo, revolucionario y novedoso en su texto como en su proceso de elaboración. Protege a niños, niñas y adolescentes, les reconoce derechos a las personas adultas mayores y en situación de discapacidad, visibiliza y reconoce derechos a sectores vulnerables, condena la violencia familiar y establece herramientas para los que han sido víctima de ella, condena la discriminación contra la mujer, democratiza las relaciones familiares, le otorga efectos jurídicos al afecto, y reconoce en su articulado la diversidad de realidades que existe entre las familias cubanas.

1. Cargamos los modelos.

```
nlp = spacy.load("es_core_news_lg")

model = SentenceTransformer('similarity-es-evaluation')

qa_pipeline = pipeline(
        "question-answering",
        model="small-gen-answers-generation",
       tokenizer="small-gen-answers-generation")
```

2. Se pone el texto en el procesador de documentos de Spacy

```
doc = nlp(texto)
```

3. Extraemos las oraciones del texto y guardándolas en una lista

```
sentences = list(doc.sents)
```

4. Hacemos un ciclo para analizar cada oración
   Cada oración será tokenizada mediante Spacy, además de extraer las entidades nombradas con el mismo.

   Se procede a tokenizar

```
# Extrayendo verbos, formas verbales al tokenizar la oracion
        for tok in docSent:
            # print(tok.text, tok.tag_, tok.dep_, tok.pos_)
            if(tok.tag_ == "VERB" and tok.dep_ == "ROOT"):
                formaverbal = tok.text
            elif(tok.tag_ == "AUX"):
                formaVerbalAUX = tok.text
           elif(tok.tag_ == "VERB" and tok.dep_ == "acl" or tok.dep_ == "advcl"
or tok.dep_ == "relcl"):
                verbo = tok.text
```

5. Extraemos las entidades y generamos las preguntas

   Por ejemplo, entidad “PER” para las personas, “ORG” para organizaciones, “LOC” para localidades, “MISC” (Misceláneas) es para varios tipos como eventos, nacionalidades, productos, etc

   Para generar la pregunta por ejemplo si es una persona:

   ¿Quién es __? el espacio que queda después de “es” lo sustituimos por la persona extraída

   Si es para una localidad: ¿Dónde queda __? añadimos la entidad localidad extraida en el espacio después de “queda”

En el paso de arriba tokenizamos para extraer verbos o formas verbales, estás se usan para generar preguntas que lleven la palabra con esa categoría gramatical.

Si se habla de una persona y hay una forma verbal casi siempre significa la acción que realizó dicha persona por lo que una pregunta sería en caso de ejemplo:

Pablo participó en un evento para desarrolladores web.

¿Quién participó? Sería la pregunta con la forma verbal extraída.

```
if(ent.label_ == "MISC"):
                    questions.append('Qué es {}'.format(ent.text))

            elif(ent.label_ == "PER"):
                personaje = ent.text
                questions.append('Quién es {}'.format(ent.text))

            elif(ent.label_ == "ORG"):
                organizacion = ent.text
                questions.append('Qué paso en o con {}'.format(ent.text))

                 if(formaverbal != "" and organizacion != "" and personaje !=
"");
                    questions.append('Quién {}'.format(
                        formaverbal) + " en {}".format(organizacion))
                # questions.append('A quien {}'.format(formaverbal))

            if(ent.label_ == "LOC"):
                # questions.append('Que paso en {}'.format(ent.text))
                questions.append('Qué es {}'.format(ent.text))
                questions.append(
                    'Dónde queda o se localiza {}'.format(ent.text))
```

6. Se crean los vectores que serán analizados para calcular la similitud coseno. A cada pregunta y oración se le hace una limpieza eliminando palabras ruido o innecesarias llamadas “stop words” o palabras vacías que no tienen peso significativo. Una vez realizada esta acción se crean los vectores con la información (tokens con un peso o puntaje significativo) lo que hace que el cálculo sea más eficiente, estos vectores están dentro de documentos que maneja Spacy.

```
for pregunta in questionsFIX:
        docPregunta = nlp(str(pregunta))

        # << Limpiando documento >>
        docPreguntaLimpio = nlp(
            ' '.join([str(t) for t in docPregunta if not t.is_stop]))

        for oracion in sentences:
            docOracion = nlp(str(oracion))

            # << Limpiando documento >>
            docOracionLimpio = nlp(
                ' '.join([str(t) for t in docOracion if not t.is_stop]))
```

7. Luego usando la función de Spacy para calcular la similitud (**similarity()**)

   *Doc_1.**similarity**(Doc_2)*

   Comparamos los documentos, hallamos su similitud y pasamos como parámetros la oración y la pregunta más similares resultantes al modelo que generará la respuesta a dicha pregunta.

   En este caso se escogen mayores a 0.50 ya que como la pregunta al ser más corta tiene menos información que la oración los puntajes de similitud no son muy altos.

```
if(docPreguntaLimpio.similarity(docOracionLimpio) > 0.50):
                # questionsResult.append(pregunta)
                # answersResult.append(str(oracion))
                result = qa_pipeline({
                    'context': str(oracion),
                    'question': pregunta})
                questionsResult.append(pregunta)
                answersResult.append(result['answer'])
```

SALIDA

Preguntas:
['Que es Código de las Familias',
'Que es Código de las Familias',
'Que es Derecho de Familia',
'Quienes integran Cuerpo',
'Quienes integran Cuerpo',
'Que paso en o con Constitución de la República de Cuba',
'Que paso en o con Constitución de la República de Cuba',
'Quienes integran Constitución de la República de Cuba',
'Quienes integran Constitución de la República de Cuba',
'Quienes integran Constitución de la República de Cuba',
'Quienes integran Constitución de la República de Cuba',
'Que es Código']

Respuestas:
['una norma sustantiva del Derecho de Familia',
'sociedad y el Estado',
'Código de las Familias',
'tutela',
'la sociedad y el Estado',
'adopción y la tutela',
'Constitución de la República de Cuba',
'tutela',
'el pueblo cubano',
'Constitución de la República de Cuba',
'familias',
'inclusivo, revolucionario y novedoso']

- En este caso se repiten preguntas ya que respecto a ellas algunos resultados de la similitud dan mayor que **0.50 (Habría que pulir esta parte, donde incluso donde puede estar la respuesta más aceptable da menor puntaje que donde no lo está).** Al igual que la versión anterior la preguntas generadas son específicas y cerradas, pues se usa el mismo algoritmo en ese sentido. **Con otros textos con una redacción que favorezca más a este método tiene mejores resultados, aunque no del todo, puede variar, como puede pasar con las demás versiones.**