Brainstorm

- Lista de fakenews detectadas → Scrapper/crawler de las páginas web seleccionadas para analizar si son o no son fake.
- Mirar qué campos se cogen para generar el **CSV de input** → GS.
- El output también se escribe sobre GS → una lista con noticias falsas.
- Al crear el cluster conectar con un componente: Jupyter Notebook, en el que está el algoritmo de detección de fakenews.
- Ya que la base de datos generada va a ser muy grande, desde el Jupyter Notebook conectamos con Hive.
- Tras analizar cómo generar el input, **se concreta la idea** y nos centramos en un dataset de transcripciones de los políticos en mítines, debates, etc. para detectar qué datos dados son falsos y que medidas prometidas son mentira.
- **Producto final:** una página web que se nutre de una base de datos en ~~MongoDB~~ en el que cada documento es un dato falso, con el político que lo dijo, dónde lo dijo y cuándo.
- Se descarta el MongoDB ya que vamos a tener datos estructurados y nos interesa más un csv con las columnas para toda la info

Diseño del DAaaS

Definición la estrategia del DAaaS

Página web en la que se puede consultar una lista de datos falsos que se han dado en distintos mítines, debates o declaraciones de los políticos. De esta forma cualquier usuario podrá ver qué político ha dado un dato manipulado, cuándo, dónde, qué dato dió y cuál es el verdadero.

Por ejemplo, cuando en un debate dan un dato relacionado con la bajada de los impuestos en muchas ocasiones está manipulado o no es preciso. Con esta arquitectura todos estos datos se contrastan y se dice si es así o no.

Para esto tendremos dos fuentes de datos: una que será el dataset de referencia, en el que están los datos contrastados y verídicos de las fuentes oficiales. La otra base de datos serán los audios con declaraciones a contrastar.

Arquitectura DAaaS

Arquitectura basada en Google Cloud con los siguientes componentes, que en el diagrama se puede apreciar mejor cómo se conectarán:

- El input inicial son los audios en *wav format* de los mítines o declaraciones que se quieren contrastar. Más adelante, estos audios serán transcritos para que puedan ser analizados correctamente.

- **Google Storage** → Con un bucket en el que tenemos dos directorios:
  - Input → Dentro hay tres carpetas:
    - Audios → donde están los audios explicados anteriormente que se consideran el input inicial.
    - Transcripciones → esto es el resultado de un job lanzado en el cluster y que se consideran el input del algoritmo.
    - Datos de referencia. → Aquí están guardados los datos que se consideran correctos. Esta es una tarea complicada a la que se debe dedicar especial atención ya que esto determinará el output.
  - Output → lista de datos considerados fake.

- **Dataproc:**

  1. Hay un job que se lanza cada semana con el input y saca las transcripciones de los audios. Estas transcripciones se guardan en la carpeta *input/transcripciones*, que será sobre lo que se trabajará en el algoritmo del siguiente paso.

  2. El cluster está conectado a un componente adicional de código abierto: Jupyter Notebook.

     En este script usaremos Hive como herramienta para tratar la gran base de datos con la que trabajamos y poder así ejecutar el algoritmo de forma más eficaz. Este algoritmo comparará los datos del input con los datos de referencia volcados en el GS y escribirá el output en ese mismo bucket. De esta forma leemos del bucket de GS las transcripciones y los datos referencia, y escribimos el output.
     La decisión de utilizar Hive es porque el usar este data warehouse nos facilita la tarea analítica por el hecho de estar montado sobre Hadoop y poder aprovecharnos de HDFS y de MapReduce.

- **Cloud SQL** para tener una base de datos que nutrirá el backend de la **web** que se ofrece como producto final. Aquí cada dato falso se corresponderá con una row, en la que también se indican en las distintas columnas: quién lo ha dado, cuándo, dónde y cuál es el dato real.

- **Cloud Function** para sincronizar los datos de GS con Cloud SQL.

DAaaS Operating Model Design and Rollout

La **carga de los datos iniciales** que se deben procesar se hace una vez a la semana por una persona que tiene asignada dicha tarea. Esta persona debe buscar los audios de mítines, debates o declaraciones de políticos y meterlos en GS con el formato indicado: wav format. Esto debe estar asociado con la información circunstancial de los audios para poder ofrecer la mejor calidad en la web.

Se pondrá en un csv y en una de las columnas todos los audios.

Proceso:

Tras esta carga se levanta un job del clúster que hace las transcripciones. Esto se hace de forma manual por una persona encargada.

Cuando este proceso acaba, de forma automática se ejecuta el código que hay en el Jupyter Notebook y que se encargará del análisis, usando Hive.

Finalmente, ya tendremos el output en GS con un listado de datos falsos y la información asociada que se quiere mostrar en la web.

Disposición del resultado:

Se vuelca el output en CloudSQL cuando está listo en GS. Esta será la parte que interactúe con el backend de la web.

El cluster estará apagado durante toda la semana. Durante este tiempo es cuando se trabajará en los siguiente aspectos para ofrecer el mejor producto posible:

- El input.
- La base de datos de referencia. Debe haber una persona que vaya actualizando estos datos para que cuando se ejecute de nuevo el algoritmo trabaje con los datos más recientes y precisos.
- Análisis del output. Se debe comprobar que el output que se está dando es correcto. Calcular las métricas de error y precisión.
- Mejora del algoritmo. Tras analizar el resultado, se dispone de esta semana para mejorar el algoritmo en lo posible.

Desarrollo de la plataforma DAaaS.

El input/audios será en **formato csv** para tener los audios con toda la información de interés asociada.

El job que se encarga de las transcripciones: será una ASR (Audio and Speech Recognition) application -- CMUSphinx Java project → escrito en **java.**

El código de Jupyter Notebook: conectamos desde ahí con la base de datos Hive y el código estará todo el **python**.

Diagrama: