UNEGOCIOS
FEN-UCHILE

Inversiones IA generativa

Clase 02 - Creación de prompts y flujos en n8n

Sebastián Egaña Santibáñez

Enlaces del profesor

https://segana.netlify.app

https://github.com/sebaegana

https://www.linkedin.com/in/sebastian-egana-santibanez/

Clase 02

Creación de prompts: Prompt Engineering

El prompt engineering es la práctica de diseñar, ajustar y estructurar instrucciones para obtener resultados precisos, útiles y reproducibles desde modelos de lenguaje como GPT. Su objetivo es reducir la ambigüedad, guiar al modelo hacia el contexto correcto y controlar el formato y el nivel de detalle de la salida.

Elementos clave

- Contexto: proveer información relevante para que el modelo entienda la situación, el rol y los objetivos.
- Instrucción clara: decir exactamente qué se espera (explicar, resumir, generar código, comparar, etc.).
- Restricciones: especificar el tono, el formato, el idioma, la longitud o estándares requeridos.
- Ejemplos (few-shot prompting): mostrar ejemplos aumenta la coherencia y la precisión del resultado.
- Iteración: los prompts se mejoran de forma incremental, evaluando la salida y ajustando instrucciones.

Buenas prácticas

- Ser específico: evitar frases vagas como "explica esto" sin contexto.
- Ser estructurado: usar listas, bullets, pasos y roles ("actúa como…").
- Ser observable: pedir formatos verificables (tablas, JSON, ecuaciones, pasos numerados).
- Ser replicable: mantener prompts reutilizables y adaptables a nuevos casos.

Se deja este enlace para utilizar: GPT para crear prompts

Flujos en n8n

Forecast de precios de acciones

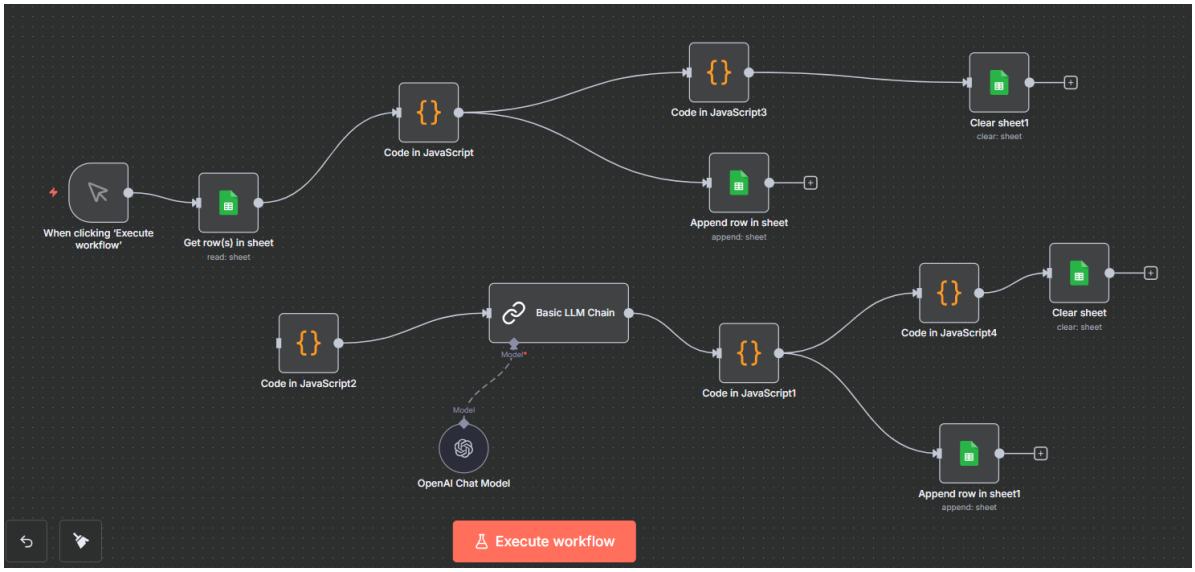


Figure 1: Diagrama

Flujo para modelo retorno promedio en base a media móvil:

1. Obtener precios: `=GOOGLEFINANCE("LTM"; "price"; "1/1/2023"; TODAY(); "DAILY")`
2. Generar proyección simple utilizando promedio de retorno por media móvil:

```
// items vienen del Google Sheets node
const rows = items;

const PRICE_COL = "Close";   // columna de precio
const DATE_COL  = "Date";    // columna de fecha en el histórico
const N = 20;                // ventana de media móvil
const H = 30;                // días a pronosticar

// 1) Extraer precios válidos
const closes = rows
  .map(r => Number(r.json[PRICE_COL]))
  .filter(x => !isNaN(x));
```

```
if (closes.length < N + 1) {
  throw new Error(`No hay suficientes datos para SMA de ${N} días`);
}

// 2) Último precio (último dato histórico)
const lastPrice = closes[closes.length - 1];

// 3) Calcular SMA de los últimos N días
const lastN = closes.slice(-N);
const smaValue = lastN.reduce((a, b) => a + b, 0) / N;

// 4) Calcular retorno promedio sobre los últimos N días
const returns = [];
for (let i = 1; i < lastN.length; i++) {
  const r = (lastN[i] - lastN[i - 1]) / lastN[i - 1];
  returns.push(r);
}
const avgReturn = returns.reduce((a, b) => a + b, 0) / returns.length;

// 5) Tomar la ÚLTIMA FECHA del histórico como base
const lastRow = rows[rows.length - 1];
const rawDate = lastRow.json[DATE_COL];  // ej: "25/07/2024 16:00:00"

// Lo forzamos a string y le sacamos la hora
const s = String(rawDate).trim();        // "25/07/2024 16:00:00"
const dateOnly = s.split(' ')[0];        // "25/07/2024"

// Partimos por / (formato DD/MM/YYYY)
const [dayStr, monthStr, yearStr] = dateOnly.split('/');

const day   = Number(dayStr);
const month = Number(monthStr) - 1;      // JS cuenta meses desde 0
const year  = Number(yearStr);

const baseDate = new Date(year, month, day);

// Validar que la fecha sea válida
if (isNaN(baseDate.getTime())) {
  throw new Error(`No se pudo convertir la fecha '${rawDate}' a Date válido`);
}
```

```
// 6) Proyección iterativa H días hacia adelante
let futurePrices = [];
let currentPrice = lastPrice;

for (let i = 1; i <= H; i++) {

  // aplicar retorno promedio
  currentPrice = currentPrice * (1 + avgReturn);

  // fecha = última fecha histórica + i días
  const d = new Date(baseDate);
  d.setDate(d.getDate() + i);

  futurePrices.push({
    date: d.toISOString().slice(0, 10),  // YYYY-MM-DD
    day_ahead: i,
    forecast_price: currentPrice
  });
}

// 7) Salida: un item por día futuro
const today = new Date().toISOString().slice(0,10);

return futurePrices.map(fp => ({
  json: {
    date_generated: today,           // día en que corriste el flujo
    forecast_type: "sma_avg_return",
    sma_N: N,
    sma_value: smaValue,
    avg_return_N: avgReturn,
    last_price: lastPrice,
    day_ahead: fp.day_ahead,         // 1, 2, 3, ..., 30
    forecast_date: fp.date,          // fecha que va aumentando
    forecast_price: fp.forecast_price
  }
}));
```

3. Llevar a un excel (Append)
4. Limpiar la hoja antes de escribir datos nuevos (Opcional)

Flujo para modelo arima utilizando LLM:

2. Extracción de columnas

```
// items viene de Get rows in sheet
const rows = items;

// 1) Extraer precios
const prices = rows
  .map(r => Number(r.json["Close"]))
  .filter(x => !isNaN(x));

// 2) Definir horizonte (por ejemplo 20)
const h = 20;

return [{
  json: { prices, h }
}];
```

3. Basic LLM Chain con modelo de Open AI: debemos introducir un prompt dentro del nodo de LLM chain.

4. Json para la construcción del forecast

```
// Parsear el JSON que llega del nodo LLM
const txt = $json.text;          // viene como string en res.text
const res = JSON.parse(txt);     // ahora es un objeto JS

const today = new Date();
const out = [];

for (let i = 0; i < res.h; i++) {
  const d = new Date(today);
  d.setDate(today.getDate() + i + 1);  // D+1, D+2, ..., D+h

  out.push({
    date_generated: today.toISOString().slice(0, 10),
    day_ahead: i + 1,
    forecast_date: d.toISOString().slice(0, 10),
    forecast_price: res.forecast[i],
    lower_95: res.lower_95[i],
    upper_95: res.upper_95[i],
```

```
    //   aquí usamos best_order y best_aic
    model_order: Array.isArray(res.best_order)
      ? res.best_order.join(',')
      : String(res.best_order),
    aic: res.best_aic
  });
}

// n8n espera un array de items { json: ... }
return out.map(o => ({ json: o }));
```

3. Llevar a un excel (Append)
4. Limpiar la hoja antes de escribir datos nuevos (Opcional)

RAG

RAG database

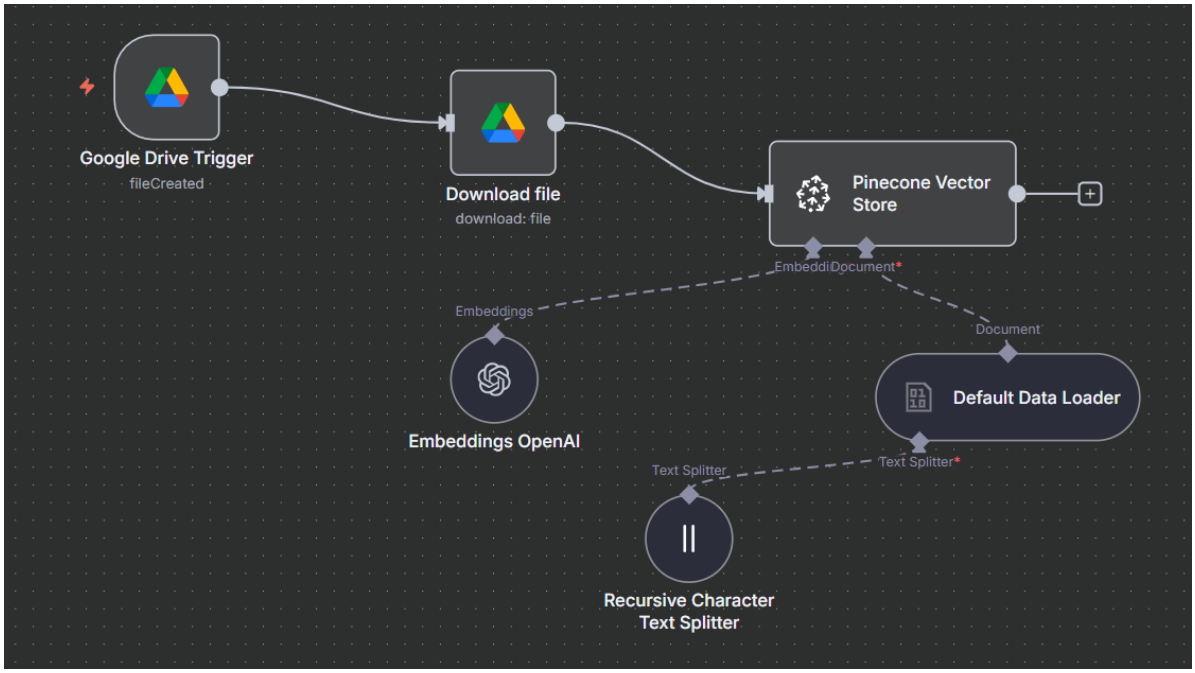


Figure 2: Diagrama

1. Trigger desde Google Drive
2. Descargar desde Google Drive

3. Pinecone Vector Store: acá se deben crear una cuenta en Pinecone (https://www.pinecone.io.

```
- Sobre batch size:
```

Es la cantidad de elementos (textos, frases, documentos, etc.) que envías al modelo de embeddings en una sola pasada o petición.

En otras palabras: Batch size = cuántos textos metes al modelo al mismo tiempo para que los convierta en vectores.

```
- ¿Por qué importa el batch size?

  - Velocidad

    - Un batch grande → más rápido porque aprovechas mejor la GPU/CPU.
    - Un batch pequeño → más lento.

  - Costo / consumo de memoria

    - Un batch grande consume más RAM o VRAM. Si usas un batch muy grande, puedes obtener: er

  - Eficiencia al usar APIs

    - Si usas OpenAI, Cohere, Mistral, etc.:

      - muchos items en un batch → menos llamadas → más barato.

      - batch muy grande → puede romper límites del proveedor.
```

4. Default data loader: opciones relevantes son elegir binary, PDF y text splitting custom. Podemos también añadir la metadata de los documentos.
5. Recurso Character Text Splitter con chunk size de 800 y un chunk overlap de 50.

Donde Chunk size es el tamaño máxomo de cada fragmento de caracteres.

Ejemplo rápido: Si tienes un texto de 3.000 caracteres y chunk_size = 800 → se divide así:

```
- Chunk 1 → 0-800
- Chunk 2 → 750-1550
- Chunk 3 → 1500-2300
- Chunk 4 → 2250-3050
```

(Dependiendo del overlap)

- Mientras más grande el chunk:

  - Más contexto tiene cada trozo
  - Más caro procesarlo (más tokens)
  - Más fácil que contenga ideas completas

RAG Agent

1. Trigger por chat
2. AI Agent
3. Chat model: OpenAI
4. Memory: Simple memory
5. Tool: Answer questions with a vector store

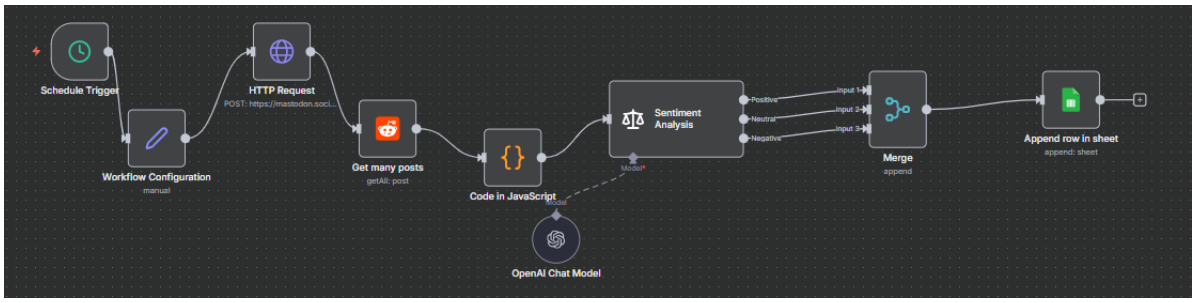
> `Returns documents relate to LATAM earning reports, outlook, margin and everything importan`

6. Vector store: Pinecone Vector Store
7. Embedding: OpenAI
8. Model: OpenAI
9. Tool: Calculadora

Modelo de sentimientos

Una mala noticia: Enlace

Veamos que debería haber sido:



Un ejemplo en Mastodon

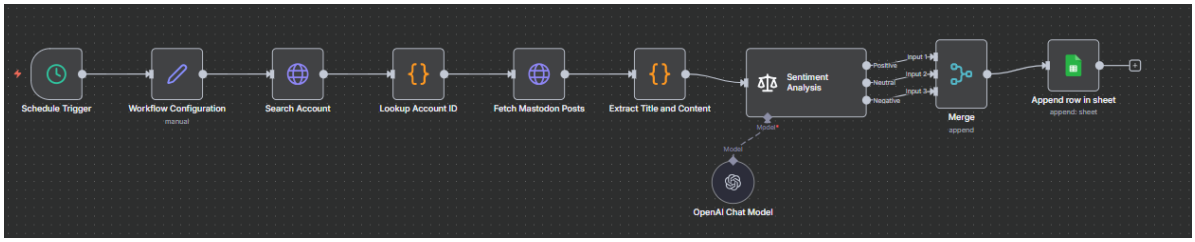


Figure 3: Diagrama

Un ejemplo con noticias de Google:

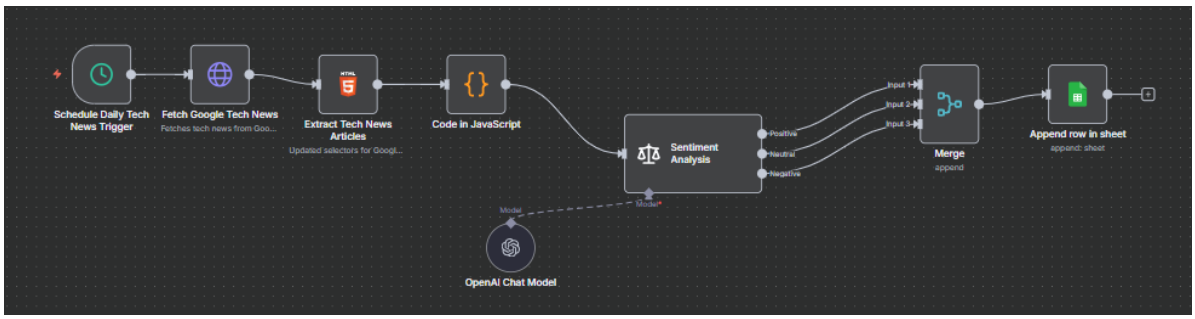


Figure 4: Diagrama