Project – Semantic Analysis

Semantic Analysis for Competency Mapping and Job Profile Recommendation

Continuous assessment

You are required to submit your solution of this assignment on BOOSTCAMP, respecting the deadline given in class. Submit your own work. Cheating will not be tolerated and will be penalized.

You may work in groups of up to 6-7 members.

Semantic Analysis

Semantic Analysis is a fundamental task in Natural Language Processing (NLP) that focuses on understanding the meaning of words, phrases, and sentences within their context. Unlike purely statistical or numerical approaches, semantic analysis goes beyond word counts or frequencies to capture how concepts are related and how they contribute to overall understanding.

In practice, semantic analysis involves:

- Word and sentence embeddings: transforming text into dense vector representations (e.g., Word2Vec, GloVe, BERT, SBERT) that preserve semantic meaning.
- Contextual interpretation: recognizing that the meaning of a word depends on its context. (e.g., Transformer model, GPT mode
- Similarity and relationship detection: using techniques like cosine similarity to measure how close two pieces of text are in meaning.

Semantic analysis is essential for applications such as:

- Text classification and clustering
- Information retrieval and semantic search
- Question answering systems
- Recommendation systems
- Competency mapping and career guidance

In this project, you will put semantic analysis into practice by comparing user-provided skill descriptions with a reference framework of competencies. Instead of relying only on numerical scores, your solution will focus on the context and meaning of words to assess coverage of competencies and recommend the most suitable job profiles.

Learning Goals:

By completing this project, students will:

- Apply text preprocessing and semantic embeddings.
- Distinguish between numerical analysis and semantic/contextual analysis.
- Learn how to design a scoring mechanism based on semantic similarity and competency blocks.
- Gain experience in building an end-to-end NLP-powered application.

- Work collaboratively to deliver and present a prototype that reflects real-world NLP use cases.

Project:
Semantic Analysis for Competency Mapping and Job Profile Recommendation

Project Description

The project gives you the opportunity to **apply semantic analysis in practice** by developing a web-based tool that helps users evaluate their skills and explore job profiles aligned with their competencies.

Objective

You will build a web application where users complete a questionnaire describing their skills and experiences with inputs of natural language type (allowing users to write descriptions in free text that can then be semantically analyzed, instead of relying only on predefined checkboxes or scores).
These inputs will then be **analyzed semantically** and compared with a **reference framework of skills and competencies**.

Key Requirements

- The **analysis must be contextual and semantic**, not purely numerical.
- User inputs will be matched against competency blocks using techniques such as:
  - word embeddings (Word2Vec, GloVe, fastText)
  - contextual embeddings (BERT, SBERT)
  - similarity measures (cosine similarity on embeddings, semantic clustering).
- A **coverage score** will be calculated using a **formula that combines multiple competency blocks** (weights, thresholds, or aggregation rules can be defined).
- The system should generate a **competency profile** for the user and suggest **job profiles(métiers)** that align with their skills.

Expected Outcomes

- A functional **web questionnaire interface**.
- A **semantic analysis engine** that compares user responses with the competency framework.
- A **competency coverage score** that aggregates alignment across multiple blocks of skills.
- A **recommendation module** that suggests the most relevant job profiles.
- **Visualization** of results (e.g., spider/radar chart, bar plots, similarity maps) to enhance interpretability.

Recommendations:

Collecting User Input (Questionnaire):

You are expected to design a web-based questionnaire **that collects user inputs through multiple types of questions**, including:

- Likert-scale questions (e.g., "Rate your proficiency in Python: 1 = beginner … 5 = expert").
- Open-ended questions (free-text answers to describe skills, experiences, or projects).
- Guided questions (structured prompts to capture specific competencies, e.g., "Have you worked with text preprocessing techniques such as tokenization?").
- Multiple-choice questions (single answer or combo-box selection).
- Checkbox questions (to allow users to select multiple relevant skills).

Web Interface Development:

To make the tool more engaging and user-friendly, you may develop an interactive user interface using:

- **Any web framework of your choice** (e.g., Flask, Django, Dash, React).
- Or **Streamlit** (recommended for quick prototypes),
  Streamlit (https://streamlit.io/) is an open-source Python framework that enables data scientists and AI/ML engineers to quickly create and deploy interactive data applications with just a few lines of code. It allows you to build powerful, dynamic apps in minutes.
  Available documentation: https://docs.streamlit.io/

The interface should:

- Be intuitive and responsive.
- Allow users to navigate between different types of questions easily.
- Collect and store responses in a structured format (CSV, JSON, or database).
- Pass the responses to the semantic analysis module for processing.

!!Attention:

The questionnaire must not just be a set of numeric ratings — it should include **contextual and semantic user input**, so that the analysis reflects meaning and competencies rather than raw numbers.
You are not just doing numbers; You are using **semantic similarity** of text embeddings to compute how well a user's profile covers the competencies in each block, then aggregating that into a meaningful **coverage score and job recommendation**.

Example of Scoring Mechanism Based on Semantic Similarity and Competency Blocks :

Below is a **concrete example** that you can consider for inspiration.
Do not forget that the analysis must remain **semantic and contextual** (not just numeric counts).

Step 1 – Define Competency Blocks

Suppose the framework has 3 blocks of competencies:

1. **Data Analysis** – {"data cleaning…", "data visualization", "statistics", "Python"}

2. **Machine Learning** – {"classification", "regression", "neural networks", "model evaluation"}
3. **NLP** – {"tokenization", "word embeddings", "transformers", "semantic analysis"}

Step 2 – Collect User Input (Questionnaire)
A user answers the questionnaire with free text:

- "I have experience in cleaning data with Python and making dashboards."
- "I studied regression models and deep learning."

Step 3 – Apply Semantic Matching

- Convert framework terms and user responses into **embeddings** (e.g., SBERT).
- Compute **cosine similarity** between user text embeddings and each competency phrase.
- Example of similarity scores you may obtain (values are illustrative):
  - User input vs. **Data Analysis block** → 0.85 (high match)
  - User input vs. **Machine Learning block** → 0.78 (medium-high match)
  - User input vs. **NLP block** → 0.40 (low match)

Step 4 – Score Calculation
Define a simple formula to combine block-level scores into a global score:

$$\textbf{\textit{Coverage score}} = \frac{\sum_{i=1}^{n} \mathrm{Wi} * \mathrm{Si}}{\sum_{i=1}^{n} \mathrm{Wi}}$$

where

- Si = similarity score for block i
- Wi = weight for block i (default = 1 if equal importance, or adjusted if some competencies are prioritized)
  Example:
  - Data Analysis: S1=0.85, W1=1
  - Machine Learning: S2=0.78, W2=1
  - NLP: S3=0.40, W3=1

$$\textbf{\textit{Coverage score}} = \frac{0.85 + 0.78 + 0.403}{3} = 0.68$$

Step 5 – Job Profile Recommendation

- If **Coverage Score ≥ 0.7** → "Data Scientist"
- If **Coverage Score between 0.5 and 0.7** → "ML Engineer"
- If **Coverage Score < 0.5** → "Entry-level Analyst"

How to Form your Reference Data:

1. **Competency Blocks**
   - Each block is a category of related competencies.
   - Inside each block, list multiple competencies/skills expressed in short phrases.
   - Example blocks: Data Analysis, Machine Learning, NLP, Project Management.
2. **Job Profiles**
   - Each job profile is linked to one or more competency blocks / or to multiple competencies.
   - A job profile is represented by the set of competencies required.

- Example: Data Scientist requires coverage in Data Analysis + Machine Learning + NLP.

3. **Why this format?**
   - Ensures the system can **compare user input** to well-structured semantic targets.
   - Allows a scoring mechanism to compute **coverage** (how much of each block is met).
   - Makes it easy to extend/update the framework with new competencies or new job profiles in the future.

Here is an example:

In real-world frameworks, a **single competency** (e.g., Python programming) can be relevant to **multiple job profiles** (Data Analyst, ML Engineer, Data Scientist, NLP Engineer).

Reference Data Structure:

CompetencyID	Competency	BlockID	BlockName
C01	data cleaning	1	Data Analysis
C02	data visualization	1	Data Analysis
C03	Python programming	1	Data Analysis
C04	regression	2	Machine Learning
C05	neural networks	2	Machine Learning
C06	tokenization	3	NLP
C07	transformers	3	NLP

JobID	JobTitle	RequiredCompetencies
J01	Data Analyst	C01; C02; C03
J02	ML Engineer	C03; C04; C05
J03	Data Scientist	C01; C02; C03; C04; C05; C06; C07
J04	NLP Engineer	C03; C05; C06; C07

For example, here Python programming (C03) is part of:

- Data Analyst (J01)
- ML Engineer (J02)
- Data Scientist (J03)
- NLP Engineer (J04)

So let's say that the user strongly mentions Python programming, the coverage score will increase in all job profiles that include it.

You may use standard skills and competencies frameworks to create your reference dataset such as :

https://www.data.gouv.fr/datasets/repertoire-operationnel-des-metiers-et-des-emplois-rome/

https://assets.ide-conseil-webmarketing.fr/wp-content/uploads/2019/05/European-e-Competence-Framework-3.0_FR.pdf

You can as well work in collaboration with your classmates to create the reference dataset.

Python Pseudocode Example: Semantic Scoring with SBERT

Here is a Python pseudocode example showing how you can implement a semantic scoring mechanism with SBERT embeddings + cosine similarity + scoring formula.

Use it only for inspiration. Customize your own solution

```
# === Step 1: Import libraries ===
from sentence_transformers import SentenceTransformer, util
import numpy as np

# === Step 2: Define competency framework (blocks) ===
competency_blocks = {
    "Data Analysis": ["data cleaning", "data visualization", "statistics", "Python"],
    "Machine Learning": ["classification", "regression", "neural networks", "model
evaluation"],
    "NLP": ["tokenization", "word embeddings", "transformers", "semantic analysis"]
}

# === Step 3: Collect user inputs (from questionnaire) ===
user_inputs = [
    "I have experience cleaning data with Python and building dashboards.",
    "I studied regression models and deep learning."
]

# === Step 4: Load SBERT model for embeddings ===
model = SentenceTransformer("all-MiniLM-L6-v2")

# Encode user inputs
user_embeddings = model.encode(user_inputs, convert_to_tensor=True)

# === Step 5: Calculate semantic similarity for each block ===
block_scores = {}

for block, competencies in competency_blocks.items():
    # Encode competency block phrases
    block_embeddings = model.encode(competencies, convert_to_tensor=True)

    # Compare each user input to competencies using cosine similarity
    similarities = util.cos_sim(user_embeddings, block_embeddings)

    # Take max similarity per user input and average across inputs
    max_similarities = [float(sim.max()) for sim in similarities]
    block_score = np.mean(max_similarities)

    block_scores[block] = block_score
```

```python
# === Step 6: Weighted scoring formula (all equal weight here) ===
final_score = np.mean(list(block_scores.values()))

# === Step 7: Recommend job profile based on thresholds ===
if final_score >= 0.7:
    recommendation = "Data Scientist"
elif final_score >= 0.5:
    recommendation = "ML Engineer"
else:
    recommendation = "Entry-level Analyst"

# === Step 8: Display results ===
print("Block scores:", block_scores)
print("Final coverage score:", round(final_score, 2))
print("Recommended job profile:", recommendation)
```

Evaluation Criteria

Your project will be evaluated on:

1. **Functional Solution**
   - Working implementation of the web-based questionnaire.
   - Correct application of semantic analysis and scoring mechanism.
   - Ability to provide job recommendations based on user input.
2. **Deliverables**
   - Source code + documentation (inline in Jupyter/Colab notebooks).
   - If documentation is separate, submit as a PDF document.
   - Clear explanations of your design choices, algorithms, and libraries used.
   - Your code must include internal comments at every step for interpretation and explanation.
3. **Presentation & Demo**
   - Each group will present their solution (slides/Notebook + demo of the application).
   - Highlight the semantic analysis part (not just numeric scoring).
   - Discuss challenges faced and how you solved them.

You have to submit the deliverables on BOOSTCAMP:

- Source Code (Python scripts / Jupyter notebooks) including documentation (group grading)
- Project presentation & Demo in class (all group members should participate; individual grading)

GOOD LUCK!