**Part A**

Solving this problem involved delicate balance between:

1. **High accuracy -** considered complex models like SBERT Bi-Encoders or BERT with Cross-Encoders, even if they're computationally expensive
2. **Data characteristics -** For smaller datasets, consider data augmentation techniques or fine-tuning smaller models like MiniLM.
3. **Supervised/Unsupervised –** supervised data has better chance of bringing in front a computationally light model as the models can be trained and tested thoroughly and appropriate tuning can be made.
4. **Available resources -** Prioritize smaller and faster models like MiniLM or simpler approaches like Jaccard.
5. **Task - Pairwise comparisons -** Most models excel at comparing two sentences

Experimenting with different options and evaluate their performance on your specific data and task.

List of Algorithms and their properties kept in mind while trying to select a model -

|  |  |
| --- | --- |
| Jaccard Similarity | * Jaccard can be a good starting point for STS, its limitations in capturing nuances of meaning and word relationships can lead to inaccurate comparisons. * Easy to understand and implement, computationally inexpensive. * **Not sensitive to word order and phrasing and Ignores word relationships.** |
| Levenshtein Distance | * Measures the minimum number of edits needed to transform one text into another. * **Focuses on surface similarity.** * **Uninformative for longer texts.** |
| TF-IDF | * Term frequency–inverse document frequency (TF-IDF) is a well-established method for measuring the importance of a word in a document, often used for information retrieval and text mining. * **Domain-independent** |
| **Cosine Distance with GloVe** | * GloVe (Global Vectors for Word Representation) is a method of obtaining word embeddings that capture semantic relationships between words. * Able to differentiate between different senses of a word * GloVe embeddings are trained on specific corpora * Less reliable for data with a large number of unique words |
| Word Movers Distance with Word2Vec | * WMD considers the "distance" between words in the embedding space, incorporating semantic relationships * WMD allows for "movement" of words between sentences * **Word2Vec is Sensitive to context** * **Computationally expensive** |
| Word Movers Distance with FastText | * FastText captures subword information, allowing it to represent and compare rare words and morphological variants more accurately than Word2Vec * **Increased computational cost** * FastText embeddings trained on biased data can inherit and amplify those biases |
| Universal Sentence Encoder | * Identifying sentences that express the same meaning in different ways * easy to integrate * trained on a massive dataset of text, which can be time-consuming |
| SBERT with Sentence Transformer (CrossEncoder) | Powerful Combination for Semantic Textual Similarity  * **Captures complex relationships** * **Domain adaptation** * **High Computational cost** |
| SBERT Bi-Encoder with Sentence Transformer | Combines the strengths of both SBERT and Bi-Encoders, leading to several advantages:**Multi-sentence similarity**Large in size, requiring more storage |
| BERT MiniLM(Deployed Model) | Smaller, Faster, and Efficient Alternative**Efficient training and inference****Wide range of pre-trained models** |

Other notable algorithms that are not covered but worth mentioning are Doc2Vec, OpenAI, Jina Embeddings, Llama models, XLNet.

**Part B**

The task provided was quite clear that it was required to deploy the Algorithm/Model built in any cloud service provider and final algorithm should be exposed as a Server API Endpoint. Initially there were various ways with which it could have been done, like Heroku, AWS lambda, AWS docker combination, AWS s3 bucket with lambda, Azure , etc. However most of these options have limitations when it comes to large models as their capacity is limited in terms of space and computational speed. One of the options that I used and was successful in deploying the Model using Fast API was with help of an AWS Ubuntu EC2 instance. It was one of the few solutions that can handle large models and provide good computational power.

Live API endpoint - <http://65.0.125.92/>