UNIVERSITY OF MILAN

MSc. Data Science and Economics

Course: Text Mining and Sentiment Analysis

HIDDEN CONCEPT

An application of Natural Language Inference

Professor: Alfio FERRARA

Student: Ly NGUYEN - 988858

Table of Content

1. Introduction

Natural Language Inference (NLI), also known as Recognizing Textual Entailment (RTE), is the task of determining whether a given hypothesis logically follows, contradicts, or remains undetermined in relation to a given premise. This task is fundamental in natural language understanding, as it plays a crucial role in various real-world applications such as question answering, machine translation, and text summarization.

However, meaning in natural language is often not explicitly conveyed through grammar, structure, or vocabulary alone. In high-context cultures, for instance, much of the intended meaning is embedded between the lines, making interpretation challenging, especially for individuals from low-context cultures. Additionally, factors such as sarcasm, word ambiguity, and multiple meanings further complicate the inference process.

This study explores hidden concepts in natural language inference by leveraging state-of-the-art large language models. By improving the ability to recognize implicit meaning, the research aims to enhance the accuracy and quality of real-world applications, including cross-cultural communication, professional interactions, and automated dialogue systems

2. Data Preprocessing

2.1. Stanford Natural Language Inference (SNLI)

“The Stanford Natural Language Inference (SNLI) corpus is a collection of 570k human-written English sentence pairs manually labeled for balanced classification with the labels entailment, contradiction, and neutral.

It is aimed to serve both as a benchmark for evaluating representational systems for text, especially including those induced by representation-learning methods, as well as a resource for developing NLP models of any kind”[1]

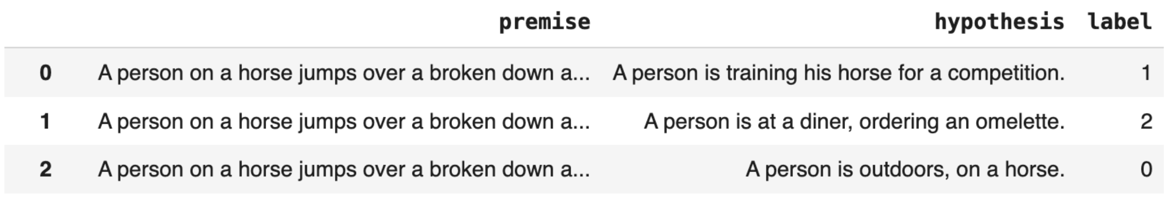
The data are sourced directly from the Python datasets library, divided into three subsets for training, evaluation, and testing. The dataset allocation is structured

[1] https://nlp.stanford.edu/projects/snli/

as follows: 80% of the data (20,000 rows) is designated for training, with the evaluation and testing sets each comprising 2,500 rows, representing 10% each.

Each dataset contains columns for premise, hypothesis, and label, all of which are fully preprocessed and encoded, making additional data cleaning unnecessary. The labels are encoded as follows:

- Entailment: 0
- Neutral: 1
- Contradiction: 2



	premise	hypothesis	label
0	A person on a horse jumps over a broken down a...	A person is training his horse for a competition.	1
1	A person on a horse jumps over a broken down a...	A person is at a diner, ordering an omelette.	2
2	A person on a horse jumps over a broken down a...	A person is outdoors, on a horse.	0

Fig1. SNLI training dataset

2.2. Enriched Stanford Natural Language Inference (Enriched SNLI)

In section 3.2, an experiment is conducted to apply pre-trained models on an enriched SNLI dataset. For this, an enriched dataset must be prepared. The approach involves using ConceptNet to retrieve relationships between word pairs, offering additional insights into the connection between premise and hypothesis sentences. These relationships are incorporated as supplementary features, aiming to enhance the model's comprehension of the data and improve performance on unseen samples.

Due to rate limits and API restrictions, the ConceptNet API limits processing efficiency, making the local ConceptNet dataset[2] a practical alternative for this experiment. This dataset underwent a careful preprocessing stage, where English-only nodes were filtered, non-essential columns were removed, retaining only the start, end, and relation columns for a cleaner, more efficient dataset.

[2] https://github.com/commonsense/conceptnet5/wiki/Downloads

3. Methodology

3.1. Pre-trained models on the SNLI datasets

3.1.1. Data Preprocessing for Transformer

The SNLI dataset was prepared for transformer models using model-specific tokenizers—bert-base-uncased, roberta-base, and distilbert-base-uncased. Each premise and hypothesis pair was tokenized to generate input IDs and attention masks, with sequences truncated or padded to a fixed length of 128 tokens. Corresponding labels were also extracted.

To speed up processing, parallelization was used, allowing each tokenizer to process the training, validation, and test datasets simultaneously. The tokenized data was then organized and converted into TensorDataset objects. Finally, DataLoader objects were created for efficient batch processing, with training datasets shuffled to improve learning. This approach ensured the data was well-prepared for model training and evaluation.

3.1.2. Model Selection

- BERT (Bidirectional Encoder Representations from Transformers)
  BERT is a highly effective pre-trained language model that has demonstrated state-of-the-art performance across a variety of NLP tasks, including Natural Language Inference (NLI). Its bidirectional architecture enables the model to capture context from both directions in a sentence, making it especially suitable for tasks like NLI, where understanding the relationship between premise and hypothesis is crucial. BERT has set a high benchmark in many NLP tasks, serving as a reliable baseline for models requiring an in-depth understanding of sentence relationships[3]
- ROBERTA (Robust Optimized BERT Approach)
  RoBERTa improves upon BERT by refining the pre-training process with additional optimizations. It is trained on larger datasets and incorporates advanced data-processing techniques such as removing the next-sentence

[3] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding

prediction task and using dynamic masking. These enhancements make RoBERTa more robust and versatile, allowing it to perform better on a wide range of NLP tasks compared to the original BERT[4]

- DISTILBERT (DISTILLED BERT)
  DistilBERT is a lighter and more computationally efficient version of BERT. By employing a model compression technique known as distillation, DistilBERT retains approximately 97% of BERT's performance while using fewer parameters. This makes it faster and more resource-efficient, ideal for scenarios with limited computational resources[5]

3.1.3. Training Set-up

BERT, RoBERTa, and DistilBERT models were fine-tuned on the SNLI dataset for three epochs. The AdamW optimizer was used with learning rates of 2e-5. A linear scheduler adjusted the learning rate during training.

Mixed precision training with autocast and GradScaler improved efficiency and reduced memory usage. In each epoch, the models processed input batches, computed and back propagated the loss, updated the optimizer and scheduler. After each epoch, validation accuracy was calculated to evaluate performance.

3.1.4. Evaluation Metric: Accuracy.

Accuracy was selected as the primary evaluation metric to assess the performance of the models. It calculates the proportion of correct predictions out of the total number of predictions, providing a clear measure of how well the models classify the relationships between premises and hypotheses in the SNLI dataset. Given the balanced nature of the dataset and the focus on overall classification performance, accuracy serves as an effective and straightforward metric to compare the effectiveness of different models.

[4] Liu, Y., Ott, M., Goyal, N., Du, J., & Kissner, M. (2019). RoBERTa: A robustly optimized BERT pretraining approach
[5] Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter

3.2. Pre-trained models on the enriched SNLI datasets

3.2.1 Enriched SNLI dataset preprocessing

As discussed in Section 2.2, an enriched version of the SNLI dataset was developed to assess how pre-trained models perform on the original dataset compared to the enriched one. Each sentence pair (premise and hypothesis) in the original dataset is processed to create a list of word pairs categorized by their part of speech—nouns with nouns, verbs with verbs, and adjectives with adjectives—while stop words are excluded from this process.

These word pairs are then matched with relationships in a local ConceptNet dataset, which provides up to 12 potential relations for the dataset as a whole. Each identified relation is added as a feature in the dataset, where each relation type is represented as a binary attribute: a value of 1 indicates the presence of a relation, while 0 indicates its absence for a given sentence pair. For example, if the first sentence pair contains the relations "synonym," "related to," and "part of," these attributes would be set to 1, while all other relation attributes would be set to 0.

3.2.2 Data preprocessing for transformer

To enhance efficiency in training and inference, we define a custom dataset class that pre-tokenized input text and incorporates additional semantic features extracted from ConceptNet. The CustomDataset class processes a given dataset by tokenizing each premise-hypothesis pair using a pre-trained transformer tokenizer, ensuring proper input representation for deep learning models. Along with tokenization, attention masks are generated to indicate which tokens should be considered in the model's computations. Additionally, several ConceptNet relationship features, such as Antonym, AtLocation, CapableOf, and Causes, are extracted and converted into tensors, providing supplementary contextual information. The dataset labels are also encoded into tensors for supervised learning. To improve efficiency, tokenized inputs and extracted features are precomputed and stored, eliminating redundant tokenization during training. The dataset class enables efficient retrieval of processed samples through the __getitem__ method, which returns a dictionary containing input tokens, attention masks, additional features, and labels.

Building upon this dataset, the CustomModel class integrates a pre-trained transformer model (such as BERT or DistilBERT) with additional semantic features to improve inference. The model is initialized using the AutoModel class from the Hugging Face library, incorporating a feature processing layer consisting of a fully connected linear transformation followed by a ReLU activation and dropout for regularization. The model's pooling strategy varies based on the transformer architecture: for models like BERT and RoBERTa, the pooled output (CLS token representation) is used, whereas for DistilBERT, which lacks a pooler, the mean of the last hidden state is used instead. The final representation is obtained by concatenating the pooled transformer output with the processed additional features, and this combined vector is passed through a classification layer to generate logits. By integrating both textual information from transformer embeddings and structured knowledge from ConceptNet, this approach enhances the model's ability to perform natural language inference tasks with improved accuracy and contextual understanding.

3.2.3 Model selection

Refer to section 3.1.2: BERT, ROBERTA, DISTILBERT

3.2.4. Training Set-up

The training and evaluation process is structured to ensure efficient model optimization and accurate performance assessment. During training, the train function is used to fine-tune the model on the dataset using mixed-precision training for computational efficiency. The function iterates over the training data in mini-batches, transferring inputs (tokenized text, attention masks, and extracted ConceptNet features) to the designated device (CPU or GPU). The model's predictions are computed, and the loss is calculated using cross-entropy loss, which is then scaled and accumulated over multiple steps before updating the model's parameters. This gradient accumulation technique helps manage memory constraints by effectively increasing the batch size without requiring excessive computational resources. The optimizer is updated using the scaler.step() method, and gradients are reset to prevent accumulation across iterations.

For model evaluation, the evaluate function assesses the model's performance on the validation and test sets. It operates in inference mode (torch.no_grad()) to disable gradient computation, ensuring efficiency. The function iterates over the evaluation dataset, making predictions and comparing them with the ground truth labels. Performance metrics such as accuracy, precision, recall, and F1-score are computed to provide a comprehensive assessment of the model's classification performance.

The main training loop initializes and fine-tunes three different transformer models—BERT, RoBERTa, and DistilBERT—by iterating through multiple epochs. The dataset is tokenized using a pre-trained tokenizer, and training and evaluation data are loaded into DataLoader objects for efficient batch processing. A custom model architecture is used, incorporating both transformer outputs and ConceptNet-derived features. The optimizer, learning rate, batch size, and other hyperparameters are configured for optimal training. After each epoch, the model's performance on the development set is evaluated, and final metrics are computed on the test set. The trained model weights are saved for future use, ensuring that the best-performing model can be deployed for downstream tasks.

3.2.5 Evaluation Metric

Refer to section 3.1.4

4. Result on Accuracy

	BERT	ROBERTA	DISTILBERT
SNLI data	82,92%	87,68%	80,05%
Enriched SNLI data	83,40%	86,48%	79,88%

5. Finding, Conclusion & Suggestion

The evaluation results show that RoBERTa achieved the highest accuracy on both the original and enriched SNLI datasets, with 87.68% and 86.48%, respectively. BERT followed with accuracies of 82.92% on the original dataset and 83.40% on the enriched dataset, showing slight improvement after enrichment. DistilBERT had the lowest performance, with 80.05% accuracy on the original data and 79.88% on the enriched data, indicating that enrichment did not enhance its performance. Overall, while data enrichment provided a minor accuracy boost for BERT, it had a limited or negative impact on RoBERTa and DistilBERT. This suggests that the effectiveness of enrichment may vary depending on the model architecture.

If resources permit, it is recommended to integrate WordNet to expand the original dataset and allow for more flexible word pairing beyond strict part-of-speech alignment. Furthermore, utilizing the ConceptNet API for direct relation retrieval would be more efficient than relying on the local dataset.