Rating Prediction from Insurance Reviews

TD Group : DIA 3
Presented By : Ahmed MAALOUL Martin PUJOL

Table of Contents

Purpose and Context

- **Context :**
  - Customer satisfaction surveys were conducted to gain a deeper understanding of client experiences. These surveys include open-ended comments, offering a valuable source of qualitative data. To analyze this data effectively, the focus is on categorizing customer feedback into meaningful groups to extract actionable insights. This process transforms unstructured text into structured information, enabling informed decision-making.

- **Objectives:**
  - Classify customer reviews into predefined categories or emerging themes.
  - Understand how these classifications relate to overall satisfaction levels.
  - Provide actionable insights to improve service quality and customer experience.

- **Techniques used :**
  - Perform an exploratory analysis of the data to uncover trends.
  - Implement classification techniques to organize feedback into meaningful themes.
  - Use text analysis methods to identify and interpret recurring patterns.

1 | Presentation of the project's objective and data

1.1 | Problem statement

1.2 | Data presentation

1.1 Problem statement

- **Understand ing Data Structure**

Transform unstructured qualitative data (e.g., customer reviews, emails, or phone call transcripts) into actionable quantitative insights.

- **Predicting Insurers' Ratings from Textual Opinions**

Enable rating predictions, and thematic analysis for decision-making.

- **Find Insight to improve insurrance opinions**

Identify meaningful keywords and themes that impact model predictions or drive sentiment.

1.2 Data presentation

Objectives

- **Clarify Feature Roles**

Explain the significance and purpose of each feature in the dataset.

- **Visualize Data Organization**

Provide a clear understanding of the data storage structure to support further analysis.

1.2.1 Data Presentation

- avis_1_traduit.xlsx
- avis_2_traduit.xlsx
- avis_3_traduit.xlsx
- avis_4_traduit.xlsx
- avis_5_traduit.xlsx
- avis_6_traduit.xlsx
- avis_7_traduit.xlsx
- avis_8_traduit.xlsx
- avis_9_traduit.xlsx
- avis_10_traduit.xlsx

- **Storage**

  - 35 files of 300 ko in average

1.2.1 Data Presentation

- **Variables**
  - Context of Notice Publication - DataTime
  - Insurer – string
  - Type (‘train’ or ‘test) ~ bool
  - Customer reviews - string
  - Translation of the notice into English - string
  - Related note - int
  - 2 None available variables (avis_ cor and avis_cor_en)

note	auteur	avis	assureur	produit	type	date_publication	date_exp	avis_en	avis_cor	avis_cor_en
5.0	brahim-k-131532	Meilleurs assurances, prix, solutions, écoute,...	Direct Assurance	auto	train	06/09/2021	01/09/2021	Best insurance, price, solutions, listening, s...	NaN	NaN
4.0	bernard-g-112497	je suis globalement satisfait , sauf que vous ...	Direct Assurance	auto	train	03/05/2021	01/05/2021	I am generally satisfied, except that you have...	NaN	NaN
5.0	virginie-t-107352	Prix tres abordable plusieurs options s'offren...	Direct Assurance	auto	train	21/03/2021	01/03/2021	Very affordable price Several options are avai...	NaN	NaN
4.0	boulain-f-116580	je satisfait du service, une réponse très rapi...	L'olivier Assurance	auto	train	10/06/2021	01/06/2021	I satisfy the service, a very fast response fr...	NaN	NaN
1.0	ouaille31-51798	Client depuis plus de 25 ans, très déçu de cet...	Matmut	auto	train	29/01/2017	01/01/2017	Customer for more than 25 years, very disappoi...	NaN	NaN

1.2.2 Translation

Translation with Google Sheets, googlestranslate function

- Everyone takes a file, demo to do
- Correcting texts to obtain correct translations

Exemple :

French input :
« Meilleurs assurances, prix, solutions, écoute, rapidité, et je recommande cette compagnie pour vous Des prix attractif et services de qualité et rapidité »
English output :
« Best insurance, prices, solutions, attentiveness, speed, and I recommend this company for you Attractive prices and quality and speed service »

Can lead to loss of information, one could think of using an embedding layer train on the French corpus…

1.2.3 Examples of observations

```
data.isna().sum()
```

	0
note	9896
auteur	1
avis	0
assureur	0
produit	0
type	0
date_publication	0
date_exp	0
avis_en	2
avis_cor	34000
avis_cor_en	34000

Focussing on test data

```
data_test.isna().sum()
```

	0
note	9896
auteur	0
avis	0
assureur	0
produit	0
type	0
date_publication	0
date_exp	0
avis_en	1

*All test data are unlabelled

*All the train are well labeled

*Translation lead to 2 NaN values

2 | Data Cleaning, Feature Engineering and Visualization

2.1 | Data Cleaning

2.2 | N-gram Distribution and Topic Modeling

2.3 | Word Embedding

* This part includes unsupervised / self- supervised approach such as W2V, Glove or DLA …

2.1 Data Cleaning

Objectives

- Identify unavailable features
- Clean the corpus to handle rare words
- Lemmatize word into list of token

Summary of cleaning work carried out

- **Preprocessing Steps:**
  - Text cleaning (removal of URLs, HTML tags, emojis, non-alphabetic characters).
  - Contraction expansion.
  - Tokenization.
  - Stop word removal.
  - Spell correction.
  - Lemmatization.
  - Each review in the dataset is converted into a list of cleaned, tokenized, and lemmatized words.

2.1.1 Simple cleaning

- **First, we only select the 'train' data, as the 'test' data is not labeled.=> Reducing data of 30 %**
- **The remaining data present only one non available translation that we dropped**
- **We decide to drop avis_cor et avis_cor_en as they are fill with Nan Value**

note	auteur	avis	assureur	produit	date_publication	date_exp	avis_en
5.0	brahim-k-131532	Meilleurs assurances, prix, solutions, écoute,...	Direct Assurance	auto	06/09/2021	01/09/2021	Best insurance, price, solutions, listening, s...
4.0	bernard-g-112497	je suis globalement satisfait , sauf que vous ...	Direct Assurance	auto	03/05/2021	01/05/2021	I am generally satisfied, except that you have...
5.0	virginie-t-107352	Prix tres abordable plusieurs options s'offren...	Direct Assurance	auto	21/03/2021	01/03/2021	Very affordable price Several options are avai...

We are just interest in the text feature and to make classification / rating / sentiment analysis from it. We might consider only keep 'note' and 'avis_en'.

2.1.2 Transformation/Cleaning of variables according to their nature

Preprocessing text :

- **Text cleaning (removal of URLs, HTML tags, emojis, non-alphabetic characters).**
- **Contraction expansion.**
- **Tokenization.**
- **Stop word removal.**
- **Spell correction.**
- **Lemmatization.**

```
# Initialize components
STOP_WORDS = set(stopwords.words('english'))

TOKENIZER = TreebankWordTokenizer()
REGEX = re.compile(r'http\S+|www\S+|https\S+|<[^>]+>')  # To remove URLs and H
EMOJI_PATTERN = re.compile("["
                           u"\U0001F600-\U0001F64F"  # emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                           u"\U0001F680-\U0001F6FF"  # transport & map symbols
                           u"\U0001F1E0-\U0001F1FF"  # flags
                           "]+", flags=re.UNICODE)
LEMMATIZER = WordNetLemmatizer()
SPELL = SpellChecker()
REGEX_NON_ALPHANUM = re.compile(r'[^a-zA-Z\s]')  # Remove non-alphabetic chara

# Optional: Define frequency thresholds for filtering
MIN_WORD_FREQ = 5
MAX_WORD_FREQ = 0.9  # As a proportion of total documents

def expand_contractions(text):
    """
    Expand contractions in the text for consistency.
    """
    contractions_dict = {
        "can't": "cannot",
        "won't": "will not",
        "n't": " not",
        "'re": " are",S
        "'s": " is",
        "'d": " would",
        "'ll": " will",
        "'t": " not",
        "'ve": " have",
        "'m": " am"
    }
    pattern = re.compile('({})'.format('|'.join(contractions_dict.keys())),
                         flags=re.IGNORECASE|re.DOTALL)

    def replace(match):
        return contractions_dict.get(match.group(0).lower(), match.group(0))

    return pattern.sub(replace, text)

def correct_spelling(tokens):
    """
    Correct spelling for a list of tokens.
    Note: Spell correction can be time-consuming for large datasets.
    """
    corrected_tokens = []
    misspelled = SPELL.unknown(tokens)
    for word in tokens:
        if word in misspelled:
            corrected = SPELL.correction(word)
            if corrected:
                corrected_tokens.append(corrected)
            else:
                corrected_tokens.append(word)
        else:
            corrected_tokens.append(word)
    return corrected_tokens
```

2.1.2 Transformation/cleaning of variables according to their nature

Expand Contractions:

- Expands common English contractions (e.g., "can't" → "cannot", "you're" → "you are") using a dictionary and regex substitution.

Correct Spelling:

Identifies misspelled words using SpellChecker and attempts to replace them with the correct spelling.

Comprehensive Text Cleaning :

- Removes URLs and HTML tags.
- Removes emojis.
- Expands contractions.
- Removes non-alphabetic characters.
- Converts text to lowercase.
- Tokenizes text using TreebankWordTokenizer.
- Removes stop words and tokens of length ≤ 1.

Tokenization-Oriented Text Cleaning:

- Corrects spelling for tokens. Lemmatizes tokens to reduce words to their base form.

2.1.2 Transformation/cleaning of variables according to their nature

Data Transformation:

Each review in the dataset is converted into a list of cleaned, tokenized, and lemmatized words.

Frequency Filtering:

Removes words that are too rare or too frequent to improve the quality of downstream tasks.

Corpus Creation:

Outputs a clean and tokenized corpus ready for use in tasks like topic modeling, classification, or embedding generation.

```
print('ORIGINAL TEXT : ',data_train['avis_en'].iloc[0])
print('TOKENIZED TEXT : ',data_train['tokenized_avis'].iloc[0])
```

```
ORIGINAL TEXT :  Best insurance, price, solutions, listening, speed, and I recommend this company for you
Attractive prices and quality services and speed
TOKENIZED TEXT :  ['best', 'insurance', 'price', 'solution', 'listening', 'speed', 'recommend', 'company', 'attractive', 'price', 'quality', 'service',
```

2.2 Data Distribution and Topic Modeling

Objectives

- **Understanding Products and Insurers Along with Their Average Ratings**
- **Understanding the Nature of the Corpus and Its Bias**
- **Visualizing Rating Distribution and Its Bias**

Summary of Work Carried Out

- **Inspect and visualize labels distributions**
- **Inspect distribution of n-gram**
- **Verification Using Zipf's Law**

Oriented Data Transformation

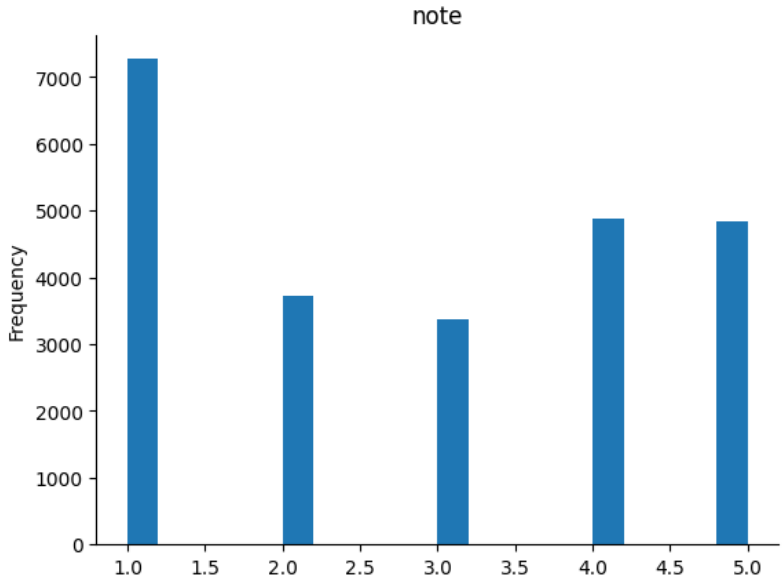
- **Frequency Filtering of Data**

Topic Modeling

- **Utilize Latent Dirichlet Allocation (LDA) for Theme Identification and Presentation**

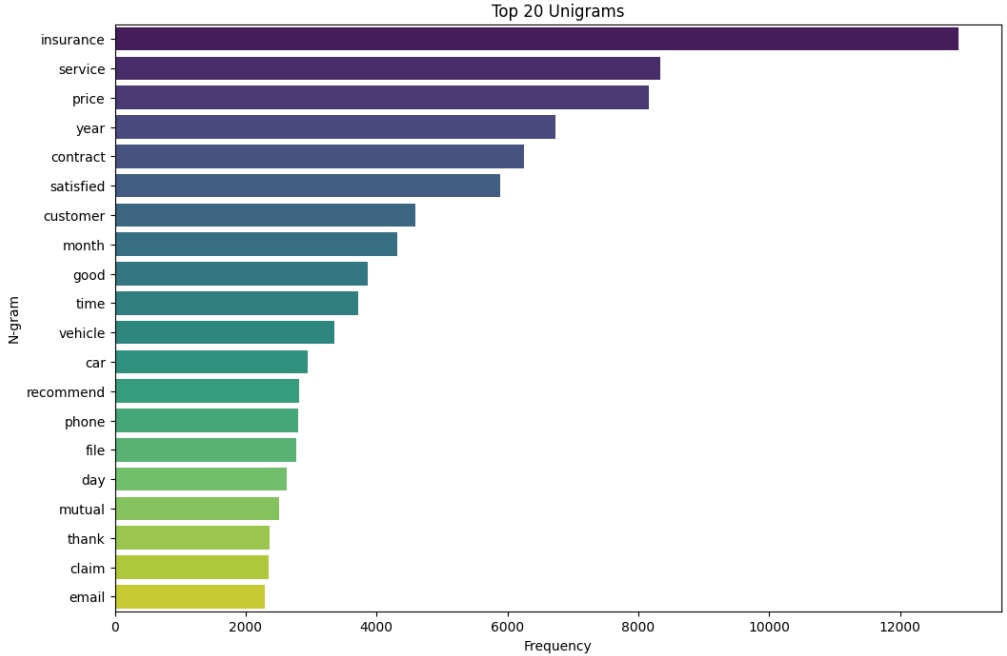
Semantic search

2.2.1 Distribution of labels



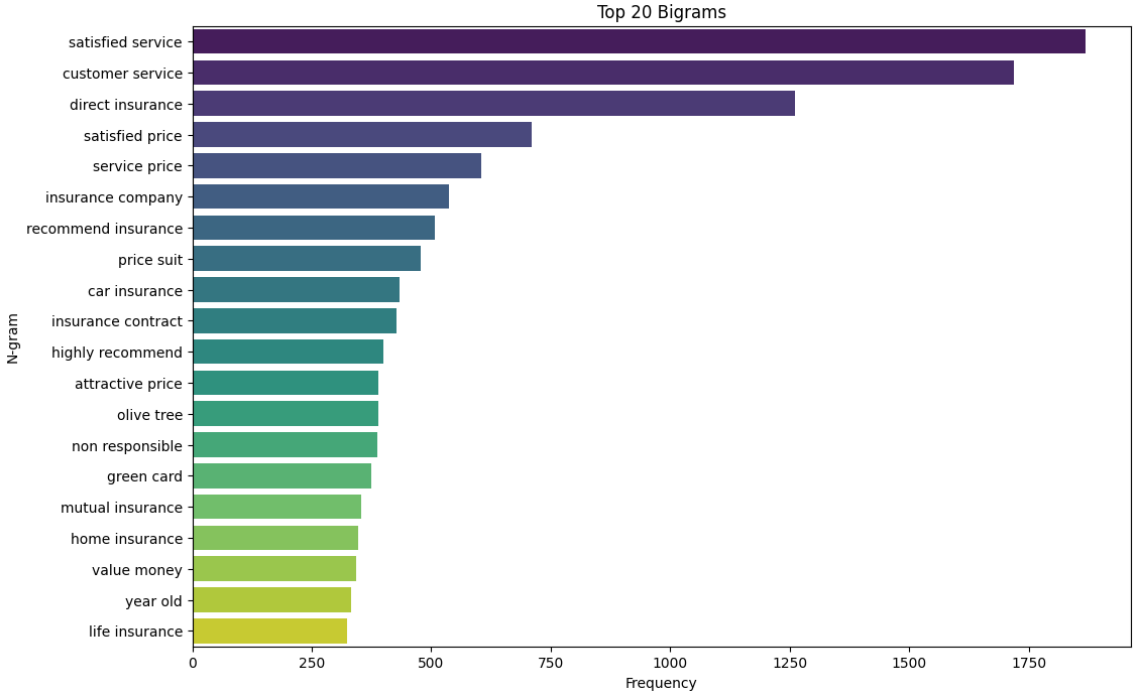
The labels appear to be well-balanced. Although the distribution is not uniform, it maintains a similar order across different classes.
We expected a strong bias toward negative experiences being reported more frequently. However, even if this is the case, the number of such reports is not even double the minimum observed for other labels, as shown in the figure.

2.2.2 N-gram Distribution



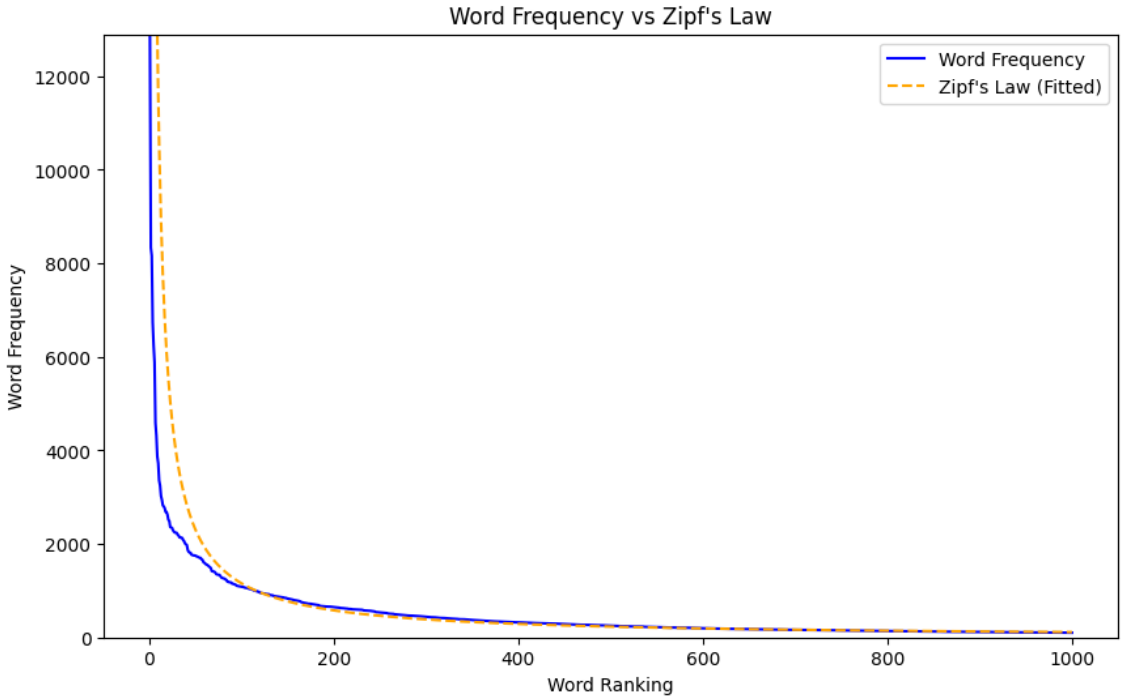
Our preprocessing enables us to filter out unnecessary and frequent tokens, such as pronouns or punctuation. However, the remaining highly frequent unigrams are still not very informative; words like 'insurance,' 'services,' and 'year' are unlikely to help us predict ratings effectively.

2.2.2 N-gram Distribution



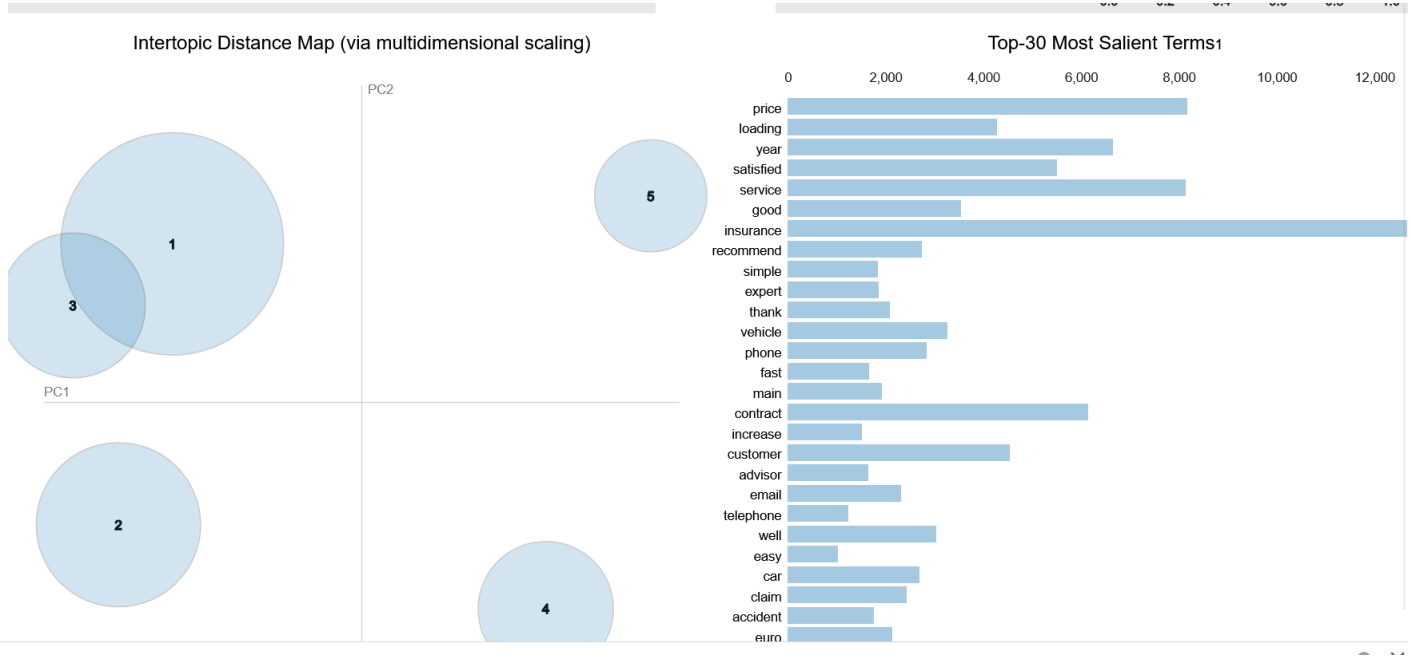
Bigrams start to give us more context as they often include an adjective rather than a pronoun, frequently conveying an opinion and thus being useful for classification.

2.2.3 Zipf’s law verification

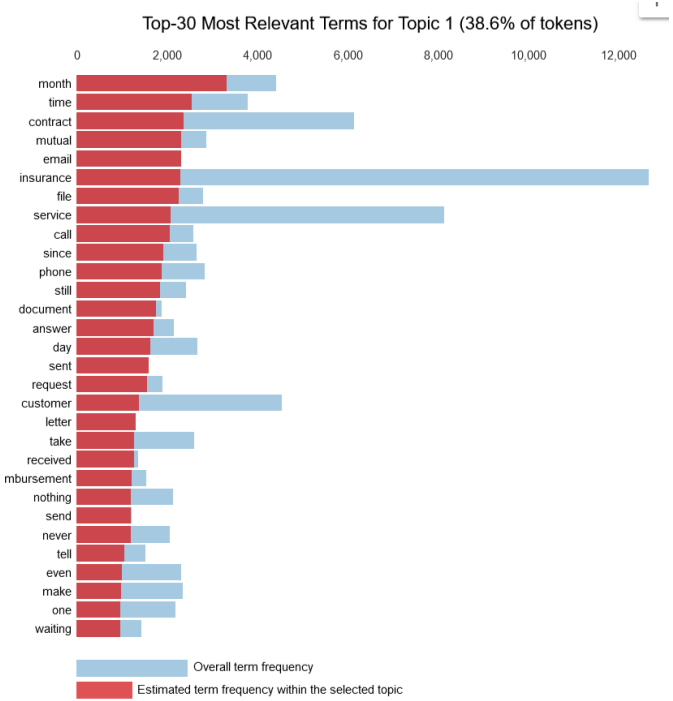


Finally, we observe a good fit with Zipf's law, confirming that useful information may be found by focusing on words that are neither too frequent nor too rare.

2.2.4 Illustration LDA - Topic modeling

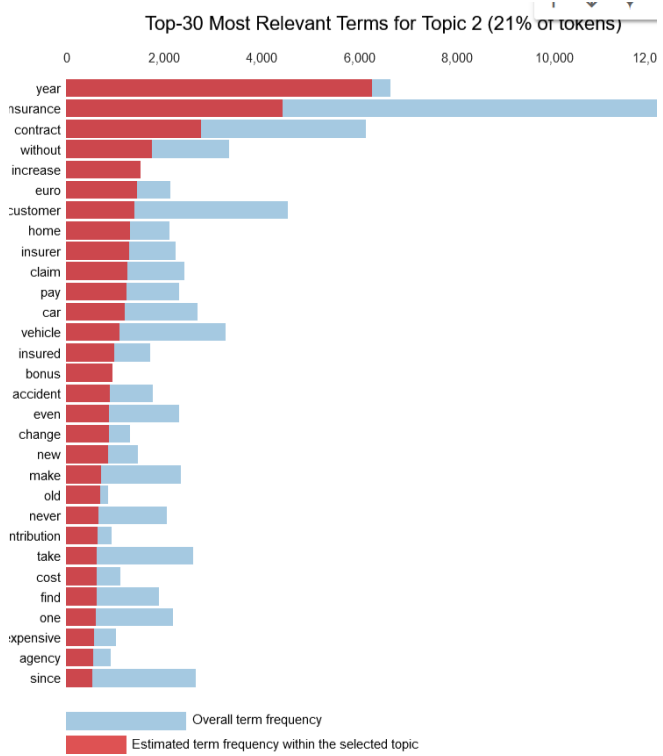


2.2.4 Illustration LDA - Topic modeling



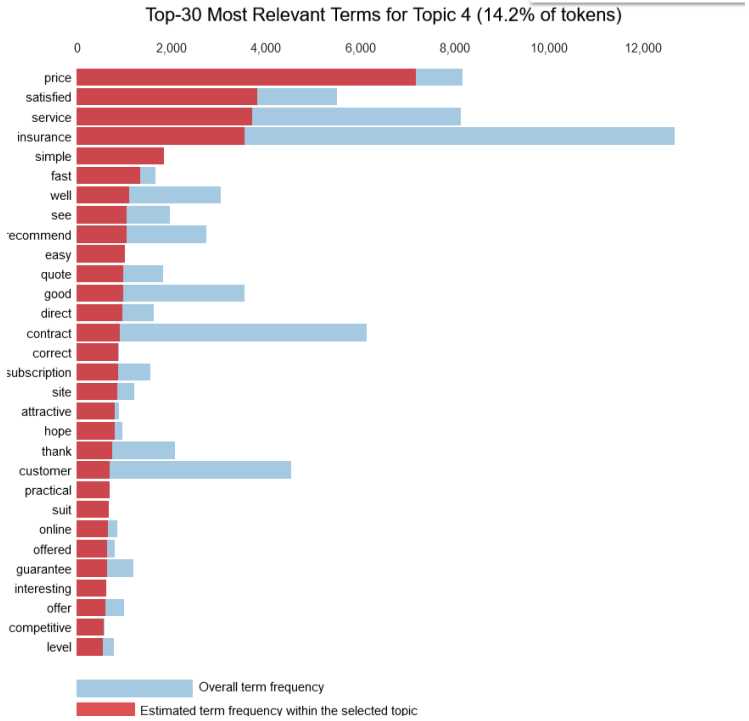
The main topic seems more relevant to administrative tasks and procedures.

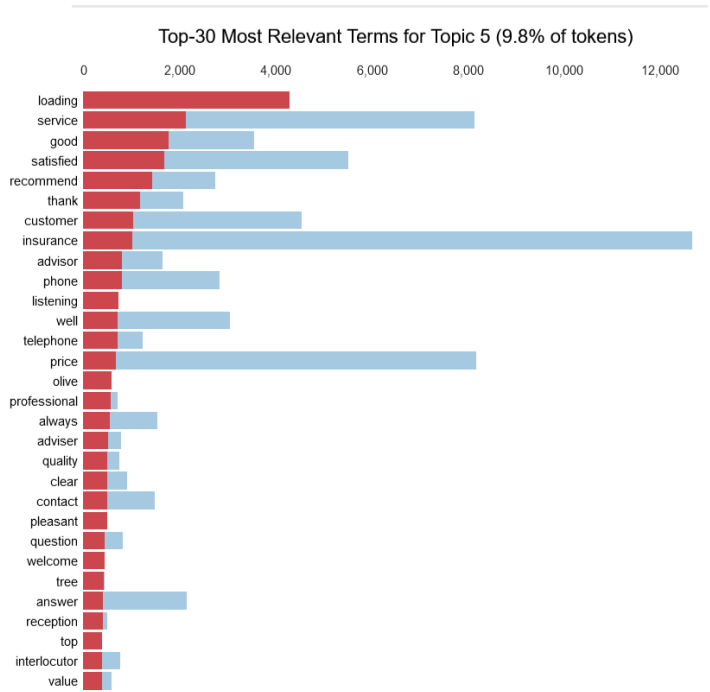
2.2.4 Illustration LDA - Topic modeling



The second one seems more related to the price of insurance. Customers seem to address the increasing price of insurance over time. We observe words such as "year," "increase," "euro," "pay," and "expenses."

2.2.4 Illustration LDA - Topic modeling





These last two topics seem related to satisfied experiences, and we might expect to see a greater overlap between them. This could be because the fifth topic appears to be related to phone calls.

2.2.5 Semantic search

```
Top documents for query: 'looking for affordable insurance'
Document Index: 2528, Similarity: 0.8222
Original Text: For a young driver I find that prices and services give by Olivier Insurance are really good. I strongly recommend this asurance to Direct Insurance whi
Tokenized  : ['young', 'driver', 'find', 'price', 'service', 'give', 'olivier', 'insurance', 'really', 'good', 'strongly', 'recommend', 'assurance', 'direct', 'insuran
--------------------------------------------------
Document Index: 21559, Similarity: 0.8140
Original Text: Listening people.
After looking for other insurers. The prices are competitive.
The contracts are clear ...
I recommend AMV insurance to those around me.

Tokenized  : ['listening', 'people', 'looking', 'insurer', 'price', 'competitive', 'contract', 'clear', 'recommend', 'am', 'insurance', 'around']
--------------------------------------------------
Document Index: 4933, Similarity: 0.8120
Original Text: After comparison with other insurers, Direct Assurance is clearly the best choice for me. Knowing that insurance comparators did not recommend me, but d
In short, compare yourself!
Tokenized  : ['comparison', 'insurer', 'direct', 'assurance', 'clearly', 'best', 'choice', 'knowing', 'insurance', 'comparators', 'recommend', 'insurer', 'offered', 'h
```

2.3 Words embedding

Objectives

- Understand the Semantic Relationships
- Explore Word Clustering
- Visualize Word embedding

Summary of Work Carried Out

- **Build Meaningful Representations:**

Develop vectorized representations of words by training a Word2Vec (W2V) model on the corpus, capturing semantic and syntactic relationships.

- **Measure Semantic Similarity:**

Implement cosine similarity to evaluate the relationships between word vectors and measure their semantic closeness.

- **Cluster and Analyze Words:**

Leverage word embeddings by clustering using k-means and cosine similarity to group similar words and uncover underlying patterns in the data.

- **Visualize Word Relationships**:

Use dimensionality reduction techniques like t-SNE to visualize the word embeddings, providing an intuitive understanding of the relationships between words.

2.3.1 Word embedding with W2V

```python
# Example: Most similar words to "insurance"
similar_words = new_model.wv.most_similar("insurance", topn=10)
print("Top 10 words similar to 'insurance':")
for word, score in similar_words:
    print(f"{word}: {score:.4f}")
```

```
Top 10 words similar to 'insurance':
assurance: 0.7184
insurer: 0.5669
newcomer: 0.5373
tree: 0.5251
ran: 0.5077
insured: 0.4993
debit: 0.4977
contract: 0.4555
hesitated: 0.4452
avanssur: 0.4420
```

2.3.1 Word embedding with W2V

Cosine Similarity

Some studies highlight the fact that word embeddings learned by models like LLMs or Word2Vec (W2V) appear to be invariant to their norms. This means that the embeddings of words typically remain unchanged when multiplied by a scalar. As a result, the most critical information for measuring their distances lies in the angle between the vectors.

For this reason, we will use cosine similarity to compute distances.

```
def cosine_sim(x1, x2):
    return((x1 @ x2) / np.sqrt((x1.T @ x1)*(x2.T @ x2)))

a = np.array([0,1])
b = np.array([1,0])

print('Cosine sim of orthogonal vectors : ', cosine_sim(a,b))
print('Cosine sim of parallele vectors: ', cosine_sim(a, a * 10))
print('cosine_sim of combinaison of parallele and orthogonal vectors : ', cosine_sim(a, (a + b) / np.sqrt(2)))

def cosine_dist(x1, x2):
    return 1 -cosine_sim(x1, x2)
```

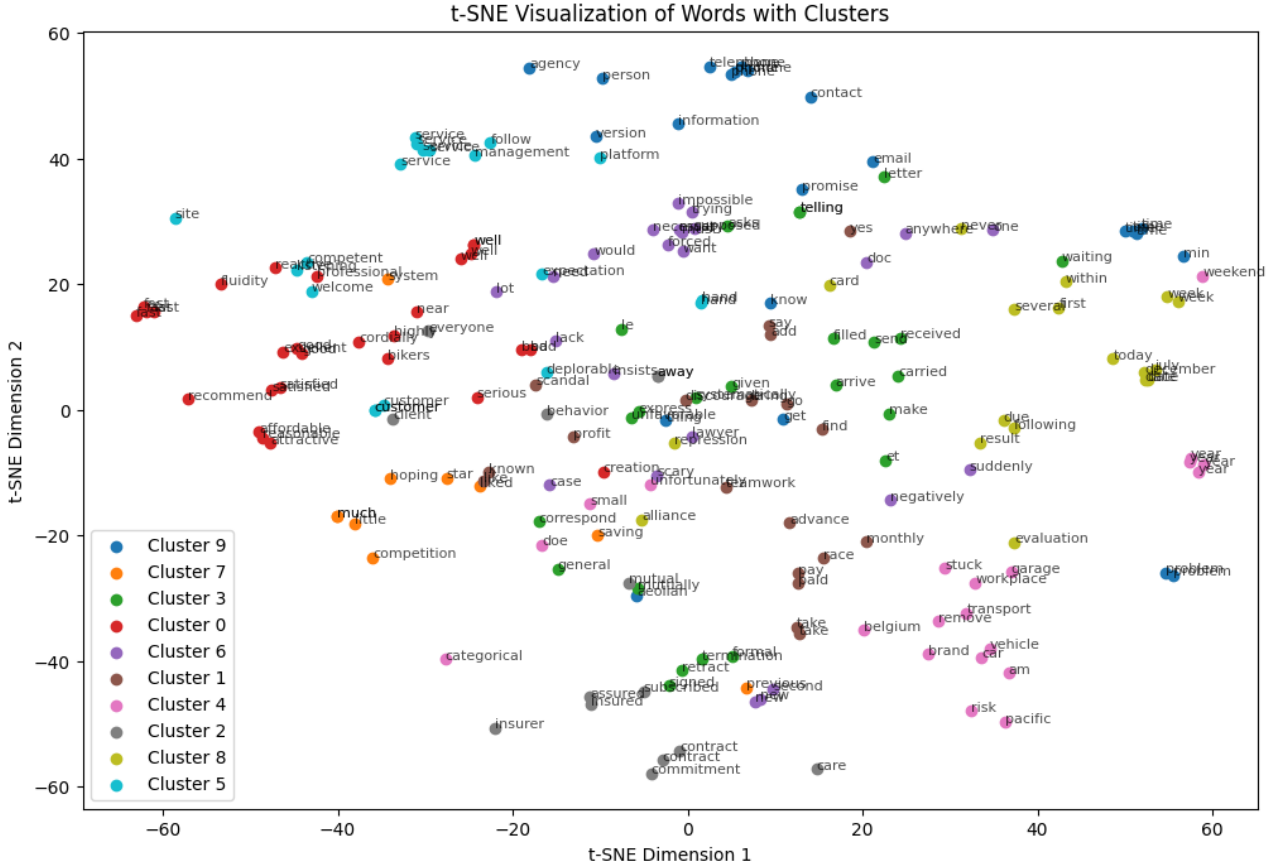
2.3.2 Clustering with k-means

We created 10 clusters using k-means, with the W2V embedding and cosine distance.

```
from sklearn_extra.cluster import KMedoids
from sklearn.metrics.pairwise import cosine_distances # Here we use the s

kmedoids = KMedoids(n_clusters=10, metric="precomputed", method='pam', )
distance_matrix = cosine_distances(group_of_vectorized_word)
labels = kmedoids.fit_predict(distance_matrix)
```

2.3.3 Embedding Visualization



2.3.3 Embedding Visualization

Analysis

Here, we can truly appreciate the power of word vectorization and how it allows us to leverage linguistic information using mathematical tools. The chosen distance metric (cosine similarity) faithfully represents the semantic distances between words. For example, we can observe that words like **contract** and **commitment** are very close to each other at the bottom of the graph. Similarly, words such as **car**, **vehicle**, and **transport** form a tight cluster, reflecting their semantic relationship related to automobiles and mobility. Another example includes **customer**, **client**, and **service**, which are grouped based on their association with business and professional contexts.

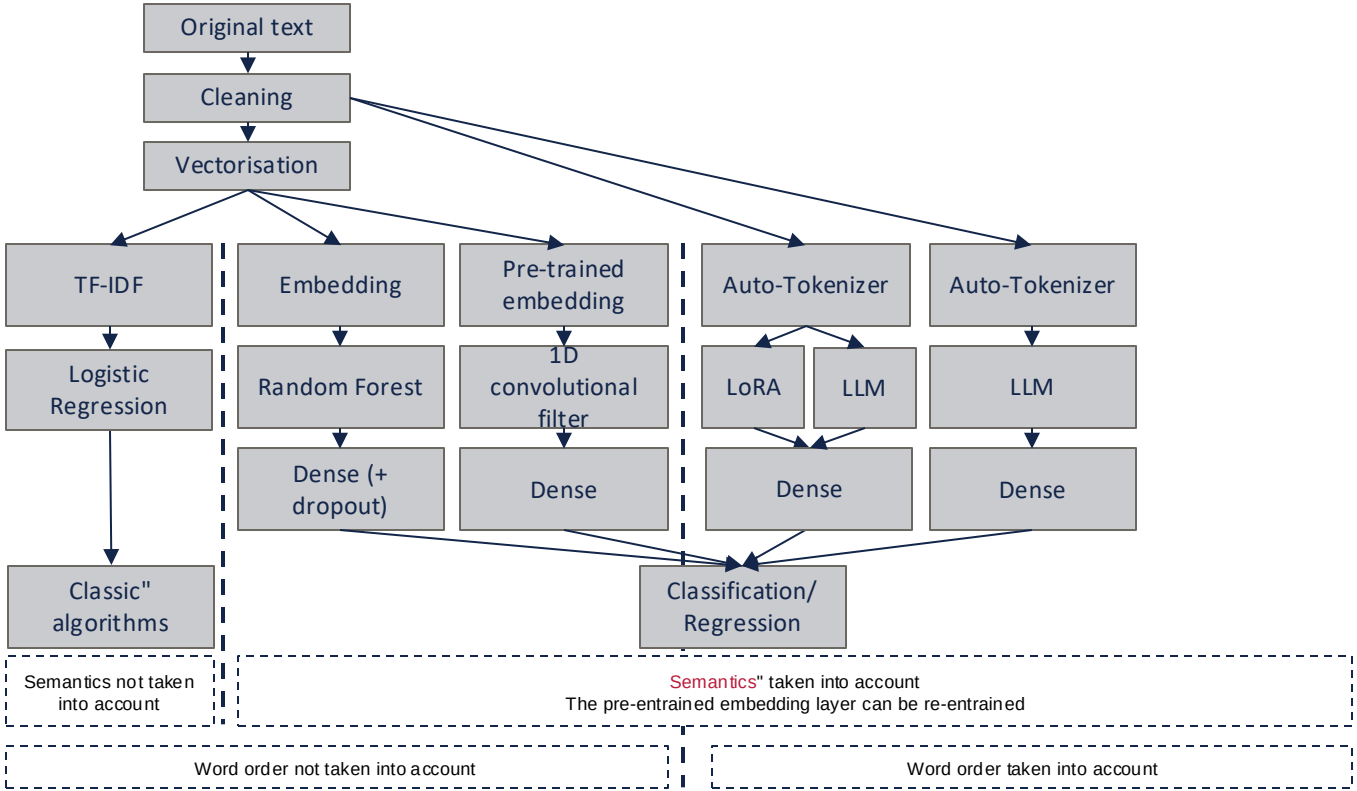
t-SNE provides a useful visualization, especially for words with very similar meanings, as it preserves similarity in the reduced-dimensional space. This clustering not only highlights semantic relationships but also demonstrates the effectiveness of word embeddings in capturing contextual meanings, making them valuable tools for tasks like information retrieval, sentiment analysis, and recommendation systems.

3 | Methodology

3.1 | Models and Pipeline

3.2 | Metrics

3.1 Models and Pipeline



3.1 Models and Pipeline

Pipeline Overview

Preprocessing and Data Splitting

Apply preprocessing steps to clean and prepare the data.

Split the dataset: 80% for training and 20% for validation/testing.

Model Training

Train the model for 10 epochs using the training dataset.

Evaluation and Metrics

Generate and display the confusion matrix to assess performance.

Calculate and present additional evaluation metrics (e.g., accuracy, precision, recall).

Visualization

Visualize embeddings, if applicable, to gain insights into feature representation.

Model Selection and Saving

Save the best-performing model based on classification average accuracy.

3.2 Metrics

- **For many of our models, we will optimize multiclass cross-entropy, which increases the separation between the distributions of data points. When evaluating predictions on an individual basis, this means assigning a distance of 1 if the predicted and true labels differ, and 0 if they are the same. Unlike regression, this approach operates differently.**

$$\text{BCE} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)\right]$$

- y_i: the true label ($y_i \in \{0, 1\}$).
- $\hat{y}_i$: the predicted probability for $y_i = 1$ ($\hat{y}_i \in [0, 1]$).
- N: the total number of examples.

3.2 Metrics

- **Finally, we evaluate the average accuracy for model selection, but we will also be very interested in visualizing the confusion matrix to see if our model struggles with different classes.**

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} 1(\hat{y}_i = y_i)$$

  - $1(\hat{y}_i = y_i)$: an indicator function that equals 1 if the prediction $\hat{y}_i$ matches the true label y_i, and 0 otherwise.
  - N: the total number of examples.

- **Additionally, we might be interested in the average absolute error, which represents the average distance between predicted labels and their targets.**

$$\text{Average Distance} = \frac{1}{N}\sum_{i=1}^{N} |\hat{y}_i - y_i|$$

  - $\hat{y}_i$: the predicted value for the i-th sample.
  - y_i: the true target value for the i-th sample.

4 | Performance Analysis

4.1 | Comparing Models performances

4.2 | Advanced Analysis of selected model

4.3 | Analysing prediction with SHAP

4.1 Exhaustive Model Performances Analysis

Objectives

- **Training Performance**: Evaluate model behavior during training to identify overfitting or underfitting.
- **Evaluation Metrics**: Assess accuracy, precision, recall, and other relevant metrics using a confusion matrix and validation data.
- **Embedding Visualization**: Explore learned feature representations through embedding visualization to gain deeper insights into model behavior.

Summary of Work Carried Out

Training and Evaluation

Trained and evaluated models including:

TD-IDF baseline model.

Basic Neural Network (NN) with an embedding layer.

Fine-tuned RoBERTa model.

Fine-tuned Llama 3.2 1B model using LoRA (Low-Rank Adaptation).

Compute Metrics

Evaluated model performance using key metrics such as accuracy, precision, recall, and F1-score.

Generated confusion matrices to assess class-specific performance.

Visualization

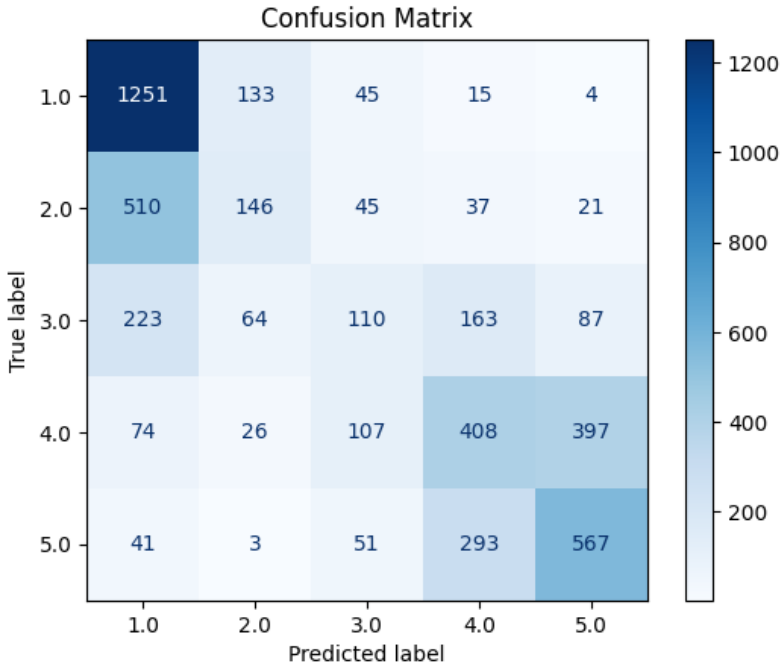
Visualized training metrics and embeddings using TensorBoard for better interpretability.

Leveraged Weights & Biases (wandb) to track and visualize novel training frameworks like LoRA.

Novel Frameworks

Implemented and applied innovative training techniques (e.g., LoRA) for large-scale model adaptation.

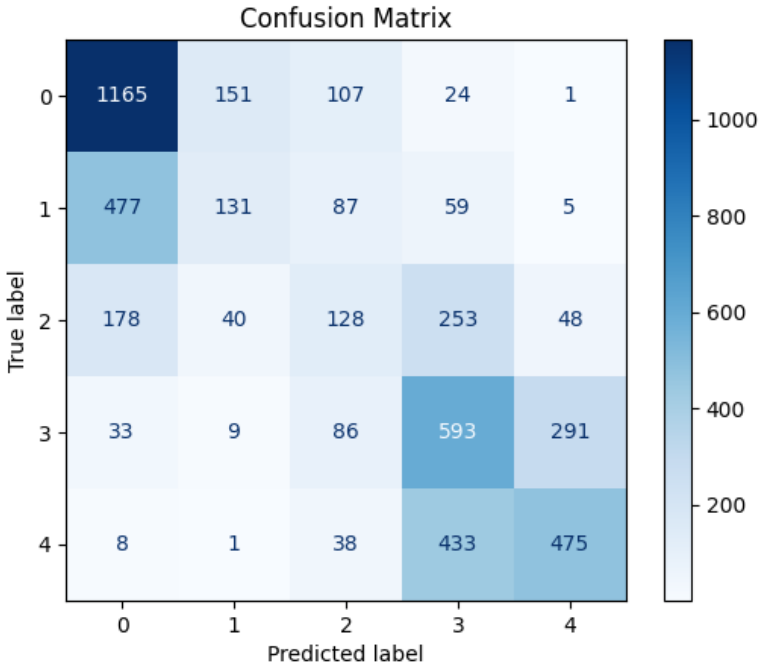
4.1.1 TD-IDF



Classification Report:

	precision	recall	f1-score	support
1.0	0.60	0.86	0.71	1448
2.0	0.39	0.19	0.26	759
3.0	0.31	0.17	0.22	647
4.0	0.45	0.40	0.42	1012
5.0	0.53	0.59	0.56	955
accuracy			0.51	4821
macro avg	0.45	0.44	0.43	4821
weighted avg	0.48	0.51	0.48	4821

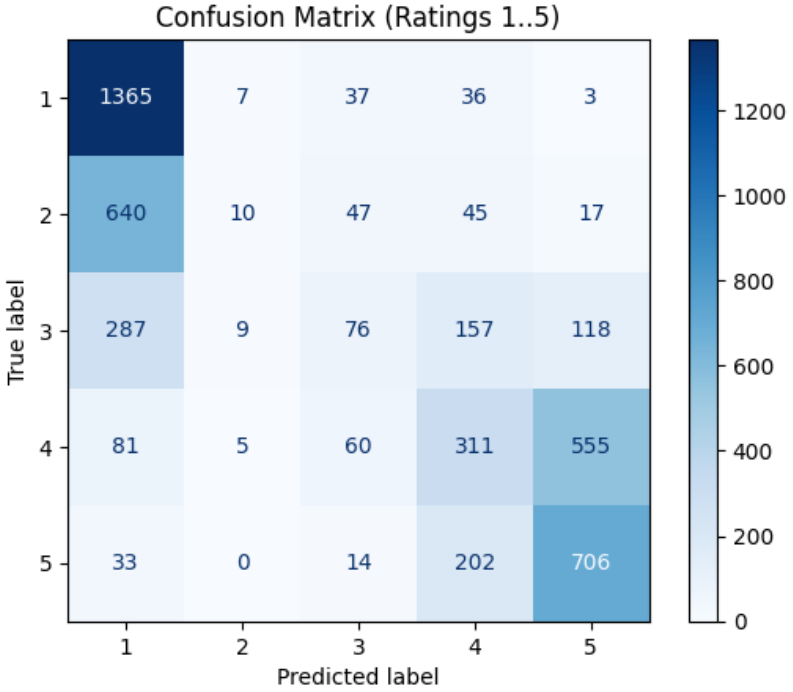
4.1.2 Basic NN (1D conv) with embedding layer



Classification Report:

	precision	recall	f1-score	support
1	0.63	0.80	0.70	1448
2	0.39	0.17	0.24	759
3	0.29	0.20	0.23	647
4	0.44	0.59	0.50	1012
5	0.58	0.50	0.54	955
accuracy			0.52	4821
macro avg	0.46	0.45	0.44	4821
weighted avg	0.49	0.52	0.49	4821

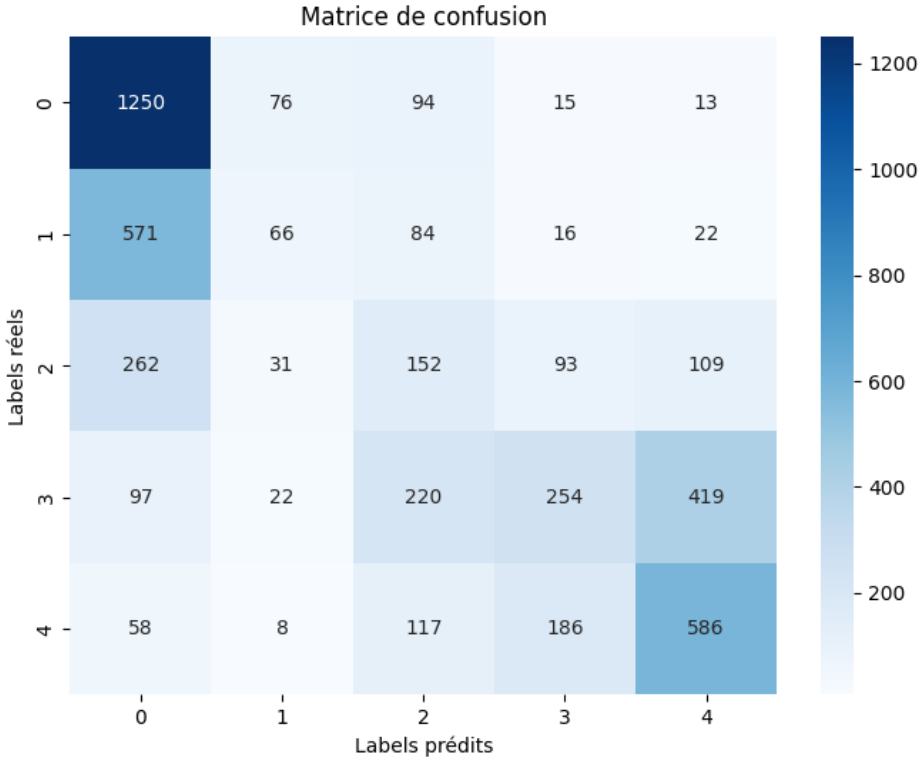
4.1.3 Fined tunned Roberta (GPT-2)



Classification Report:

	precision	recall	f1-score	support
1	0.57	0.94	0.71	1448
2	0.32	0.01	0.03	759
3	0.32	0.12	0.17	647
4	0.41	0.31	0.35	1012
5	0.50	0.74	0.60	955
accuracy			0.51	4821
macro avg	0.43	0.42	0.37	4821
weighted avg	0.45	0.51	0.43	4821

4.1.4 Fined tunned llama 3.2 with LoRa

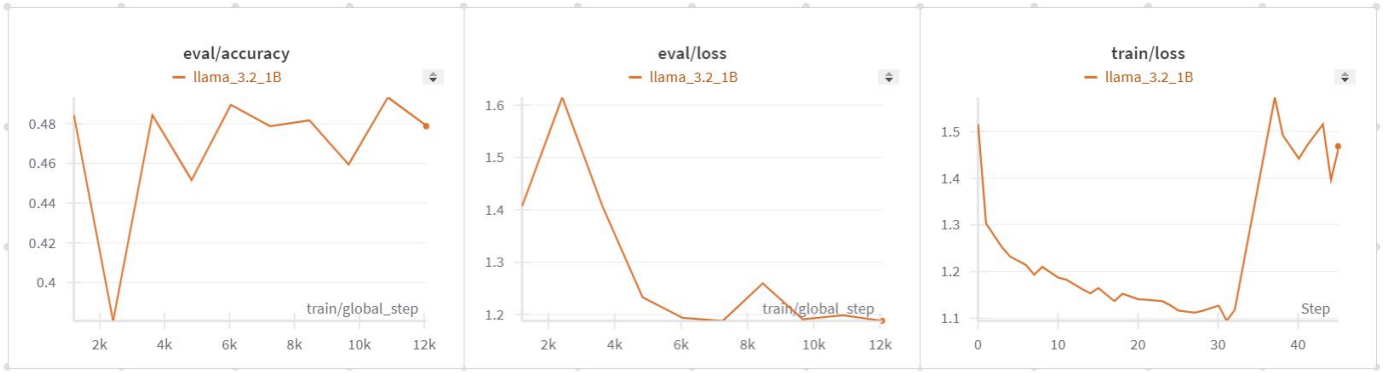



```
Detailed Classification Report:
              precision    recall  f1-score   support

     Class 0       0.56      0.86      0.68      1448
     Class 1       0.33      0.09      0.14       759
     Class 2       0.23      0.23      0.23       647
     Class 3       0.45      0.25      0.32      1012
     Class 4       0.51      0.61      0.56       955

    accuracy                           0.48      4821
   macro avg       0.41      0.41      0.39      4821
weighted avg       0.45      0.48      0.43      4821
```

4.1.4 Fined tunned llama 3.2 with LoRa



4.2 Model Selection and Error Analysis

Objectives

- **Understanding the project and its context**
- **Understanding the benefits of predictive models**

4.2.1 Model selection

After comparing performances, we decided to select the Basic NN model, as it delivered better performance than the large 1B-parameter model like Llama 3.2. Its size is less than 10 MB, making it lightweight and easy to integrate into our Streamlit application.

Layer (type)	Output Shape	Param #
embedding_layer (Embedding)	(None, 100, 64)	640,000
global_average_pooling1d (GlobalAveragePooling1D)	(None, 64)	0
dense (Dense)	(None, 32)	2,080
dense_1 (Dense)	(None, 6)	198

Total params: 642,280 (2.45 MB)
Trainable params: 642,278 (2.45 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 2 (12.00 B)

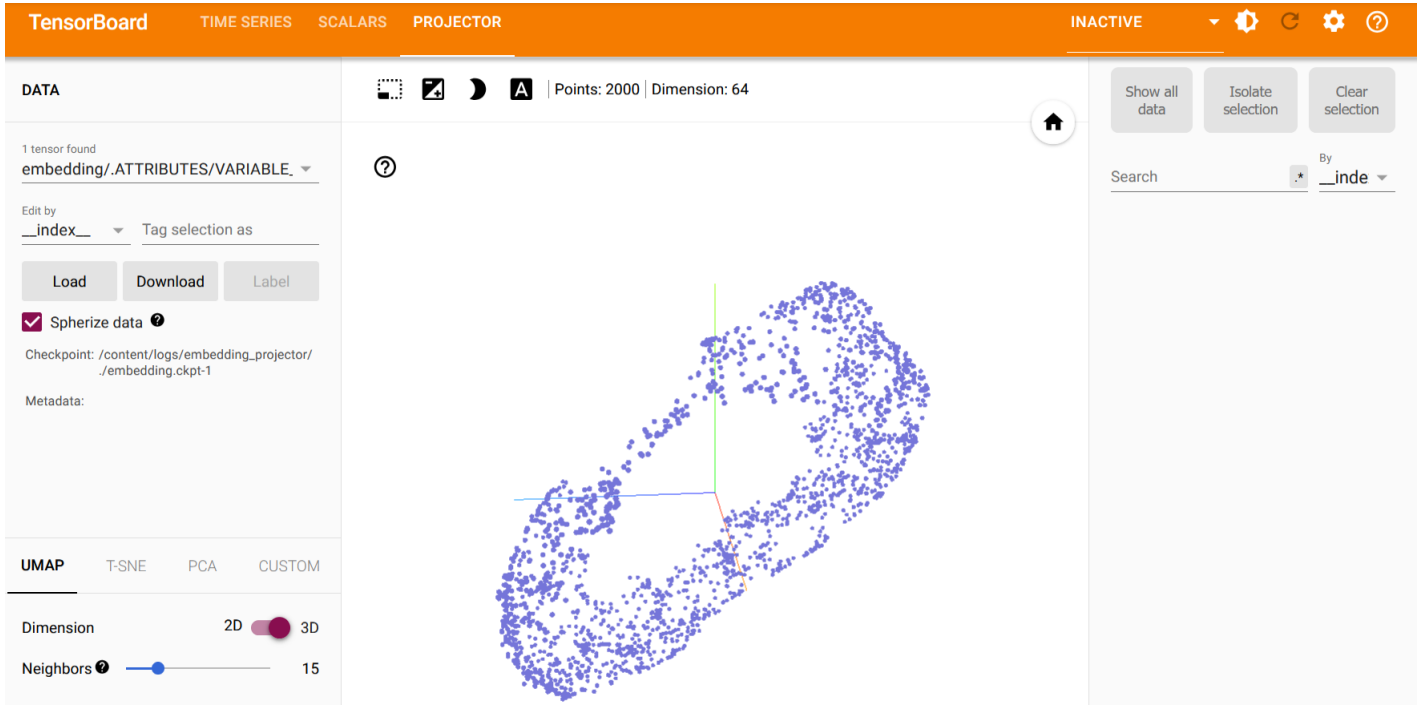
4.2.2 Model analysis

```
Mean Distance: 0.7529558182949595
Mean Precision: 0.41438138570990707
Mean Recall: 0.4099494692414999
```

A mean distance of 0.75! This means that, on average, our predictions remain close to the expected values, with a distance less than one. This conclusion can also be inferred from the confusion matrix.

4.2.2 Model embedding visualization

Try TensorBoard visualization in our notebook!



4.2.3 Error analysis

The model is really sensitive to money information, as their are more corralted to bad review n the corpus :

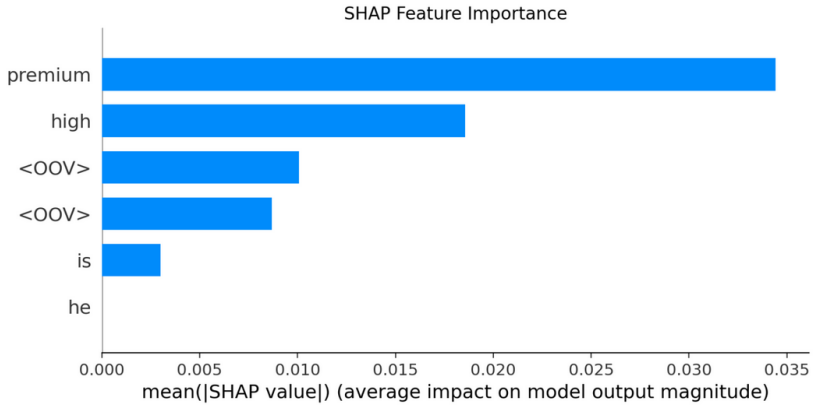
Review:

he premium is surprisingly high, but I'm grateful for the peace of mind that comes with such excellent coverage.

Predict

Prediction

Predicted Class: 1



Here premium and high oriented the model to give a bad rating

4.3 Model Interpretation

Objectives

- **Themes explaining the level of satisfaction**
- **Keywords/embeddings that explain the level of satisfaction**

4.3 Model interpretation

It is hard to establish a global framework for analyzing our model. One of the best methods is to examine predictions given an input.
This is what we can do with SHAP; here is an example:

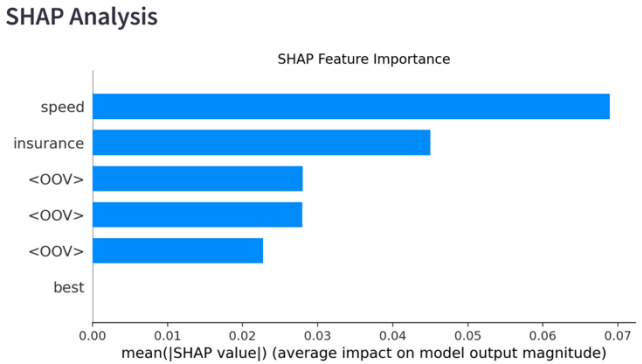
Prediction of Insurance Review Rating

Enter an insurance review below to predict its rating.

Review:

Best insurance, prices, solutions, attentiveness, speed, and I recommend this company for you
Attractive prices and quality and speed services

Predict



➔ High influence of the term « Speed »

5 | Streamlit applications

5.1 | Infrastructure

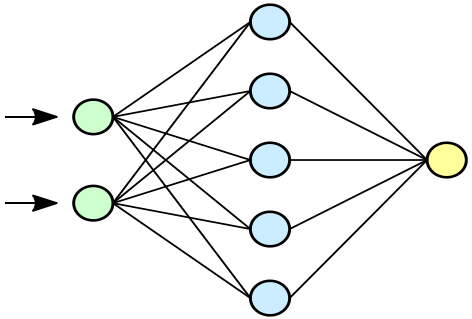
5.2 | Real-Time prediction

5.3 | SHAP analysis

Transform your opinion in a rating !
Try it there :

https://insurance-opinion-classification-3hv3edp8cpphd4dwgtwzcc.streamlit.app/

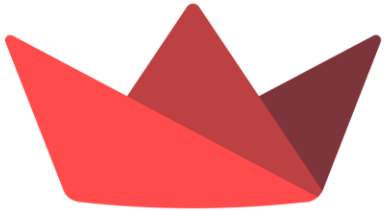
5.1 Infrastructure



Data, code
and models

GitHub

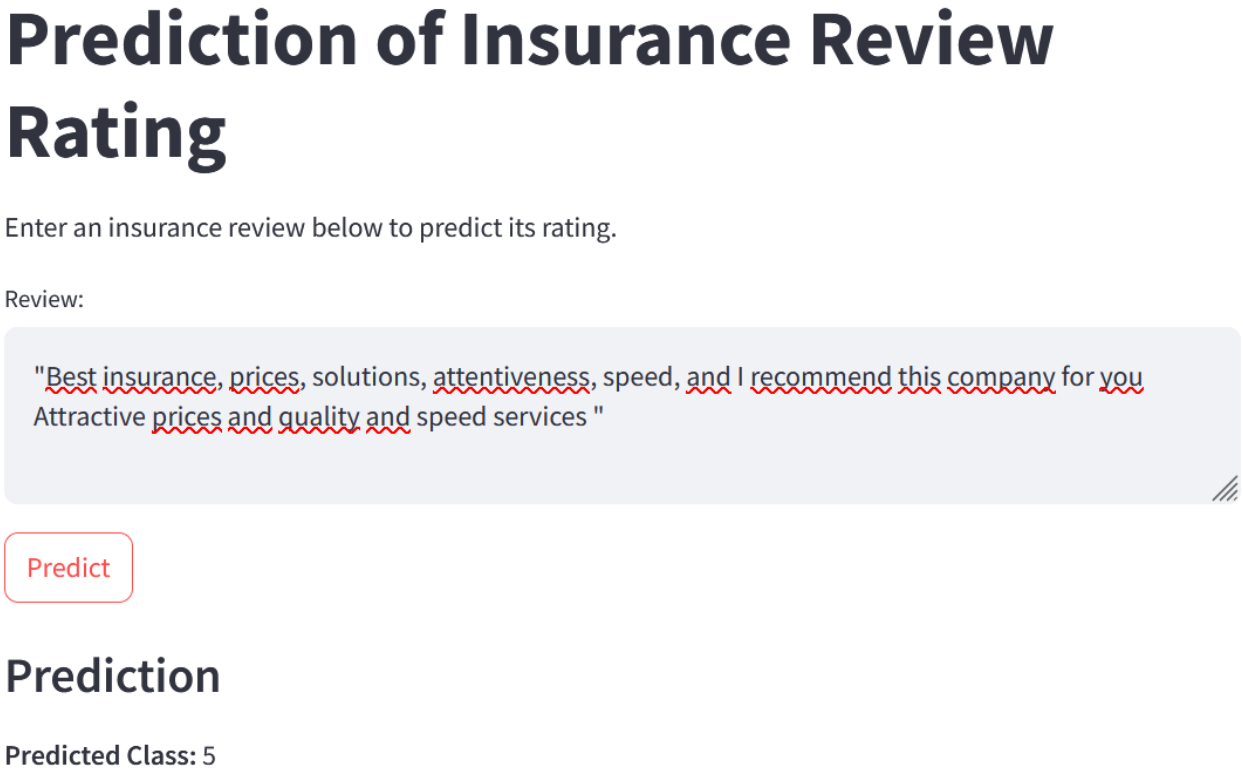
Deployment
with backends



Layer (type)	Output Shape	Param #
embedding_layer (Embedding)	(None, 100, 64)	640,000
global_average_pooling1d (GlobalAveragePooling1D)	(None, 64)	0
dense (Dense)	(None, 32)	2,080
dense_1 (Dense)	(None, 6)	198

Total params: 642,280 (2.45 MB)
Trainable params: 642,278 (2.45 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 2 (12.00 B)

5.2 Real-Time prediction



5.3 SHAP Analysis