In [36]:
```
import matplotlib.pyplot as plt
from data import possible_frequencies,possible_rarities
from test_tfidf import TestTFIDF
```

In [37]:
```
testTFIDF = TestTFIDF(possible_frequencies,possible_rarities)
```

The frequencies and rarities here are taken from the data.py module.
To use the TestTFIDF object I sent requests to my TF-IDF node service running locally.

In [38]:
```
plt.rcParams['figure.figsize'] = [12, 5]
testTFIDF.start_test({ "tf_option": 1, "idf_option": 1 })
testTFIDF.start_test({ "tf_option": 2, "idf_option": 1 })
testTFIDF.start_test({ "tf_option": 2, "idf_option": 2 })
```

TF-IDF Weighting Schemes

TF options

1. **Normal:** $f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
2. **Log normalization:** $\log(1 + f_{t,d})$

IDF options

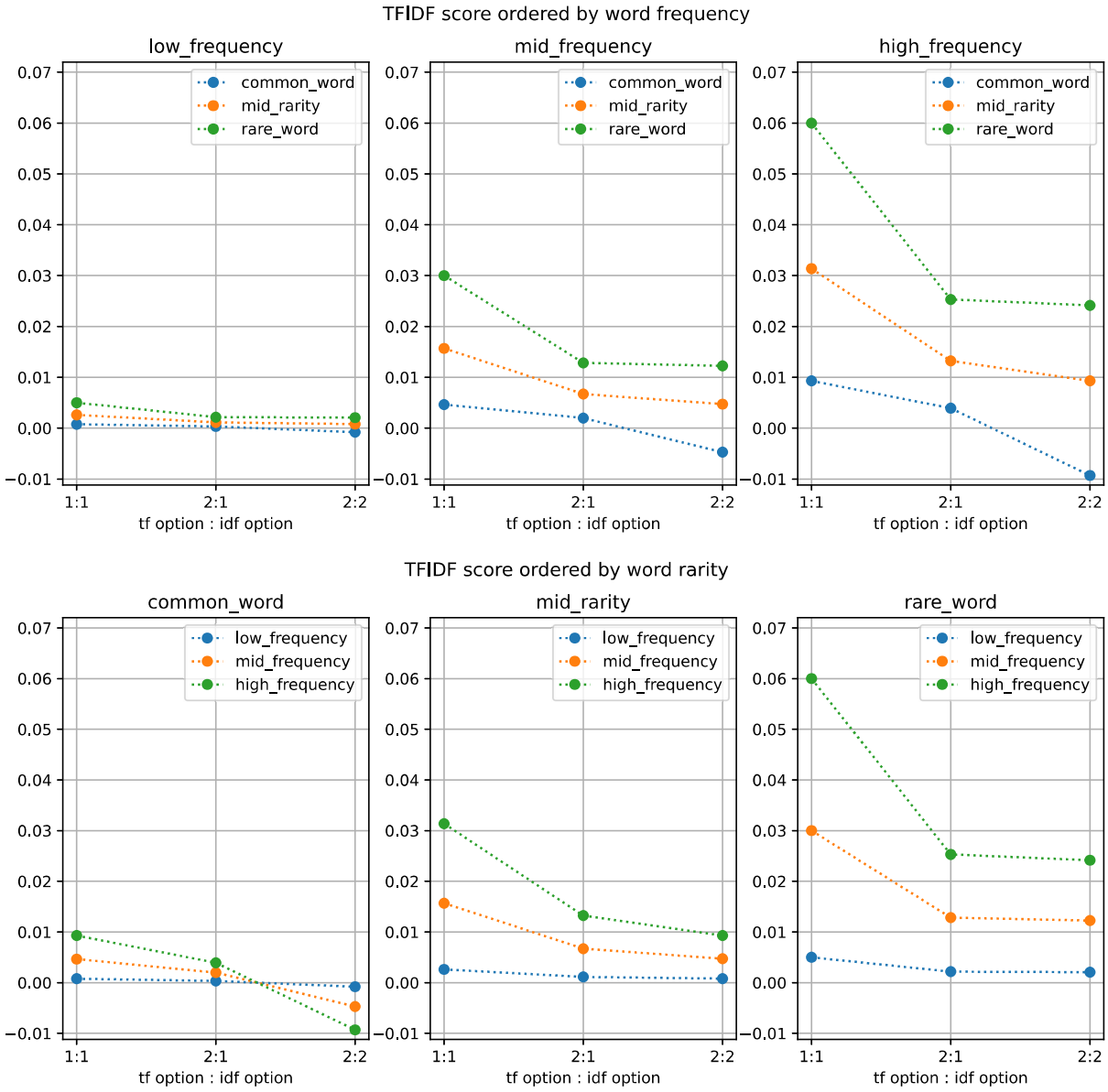
1. **Normal:** $\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
2. **Probabilistic:** $\log \frac{N - n_t}{n_t}$

Note on word's rarity:

I used the term 'rarity' here to describe the document frequency.
If more then half of the documents contains the given word, I called it a common word.
If only tenth of the documents contains it, it's a rare word.

In the following graphs, the Y axis describe the TF-IDF score, and the X axis is the weighting schemes combinations (3 lines for 3 combinations).

In [39]:
```
testTFIDF.draw_graphs()
```



I chose the following tf-idf weighting schemes combinations: (The options are explained above)

- tf_option: 1, idf_option: 1 (1:1)
- tf_option: 2, idf_option: 1 (2:1)
- tf_option: 2, idf_option: 2 (2:2)

Conclusions

We can deduce from the graphs that the third combination gives the common words a negative score (a possible usecase is to easily filter out common words, like 'the').
Also, the log version of the TF is squeezing the tf-idf score to a smaller range, as we expect from the log function. This is especially reflected in the high_frequency graph, where the tf score is relatively high.
There is no clear difference between the second and third combination, other then what I've mentioned about the negative score.