Hybrid Approaches in NLP: Combining Traditional Algorithms and Fuzzy Logic for Textual Insights

Vasyl Melnyk

TRAINING COURSE «Dealing with propaganda, misinformation and fake news using AI&ML

Swedish Institute

KNEU Science Park

ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА 1875 VIVAT ACADEMIA

Author

- **Vasyl Melnyk**
- PhD in mathematics, assistant at Chernivtsi National University, dept. of Mathematical Modelling
- Scientific Interests: nlp, fuzzy logic, ml, ai, functional analysis, topology, generative functions, enumerative mathematics, educational methodology
- Coauthors: prof. Halyna Melnyk; students Valentyn Vikovan and Andrii Lazoryk

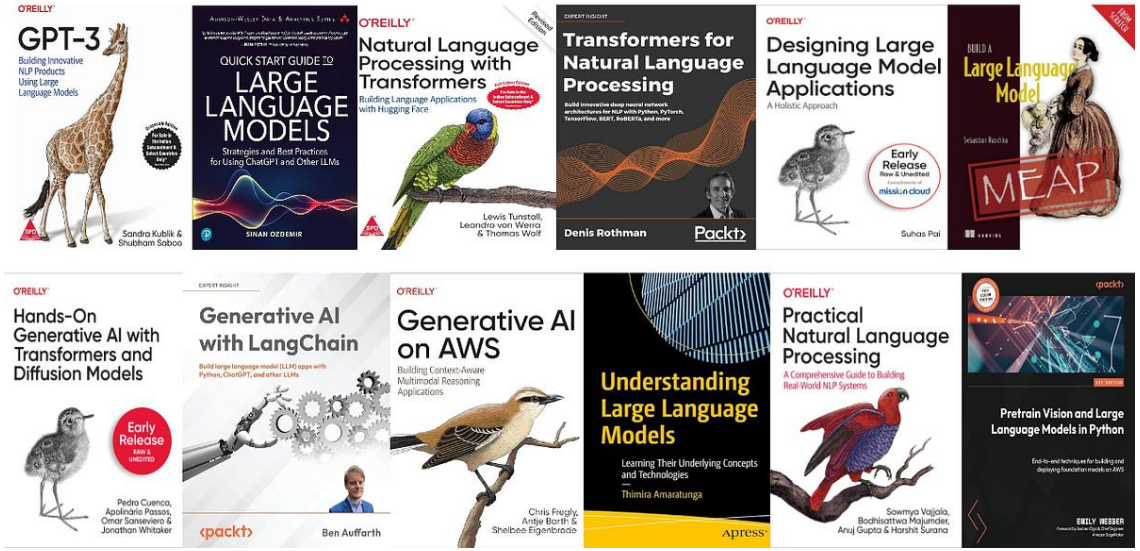
Agenda

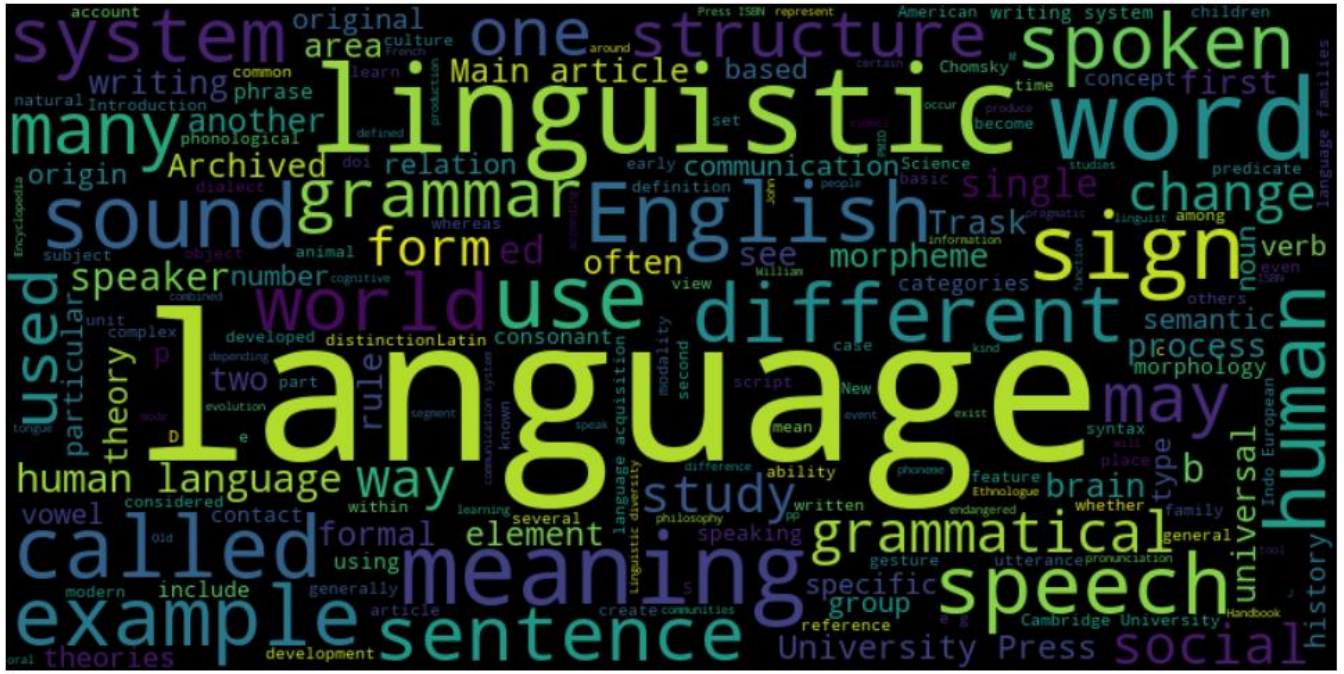
- NLP Techniques
- Fuzzy Logic
- Applications to Disinformation Text Analysis
- Text Sentiment Analysis
- Future Research Directions and Conclusion

NLP

- NLP is a subfield of artificial intelligence focused on enabling computers to understand, interpret, and generate human language.

Books on LLM and NLP in 2024





NLP: TFIDF

- Term Frequency-Inverse Document Frequency: Measures the importance of a word in a document relative to a collection (corpus) of documents.

```
documents = ["Alex likes to walk with his dog",
             "Alex and his dog are best friends",
             "hello world this is random text"]
```

NLP: TFIDF

- Let's construct a matrix that contains word counts (term frequencies) for all the documents

```
documents = ["Alex likes to walk with his dog",
             "Alex and his dog are best friends",
             "hello world this is random text"]
```

```
{'Alex': 0, 'and': 1, 'are': 2, 'best': 3, 'dog': 4, 'friends': 5, 'hello': 6, 'his': 7, 'is': 8,
 'likes': 9, 'random': 10, 'text': 11, 'this': 12, 'to': 13, 'walk': 14, 'with': 15, 'world': 16}
```

```
[[1 0 0 0 1 0 0 1 0 1 0 0 0 1 1 1 0]
 [1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 1 0 1 1 1 0 0 0 1]]
```

NLP: TFIDF

- An obvious issue with using normal term frequency counts to represent a document is that the document's vector will often be "dominated" by very common words, for example: "of", "the", "is", in the preceeding example documents. Intuitively, we expect that the most "important" words in a document are precisely those that only occur in some relatively small number of documents, so that we want to discount the weight of very frequently-occurring terms.
- This can be accomplished via the inverse document frequency weight for words. Just as with term frequencies, there are some different weightings of this term, but the most common formulation is

$$idf_j = \log\left(\frac{N_{documents}}{N_{documents\ \ with\ \ word\ \ j}}\right)$$

NLP: TFIDF

$$idf_j = \log\left(\frac{N_{documents}}{N_{documents \;\; with \;\; word \;\; j}}\right)$$

```
{'Alex': 0, 'and': 1, 'are': 2, 'best': 3, 'dog': 4, 'friends': 5, 'hello': 6, 'his': 7, 'is': 8,

    'likes': 9, 'random': 10, 'text': 11, 'this': 12, 'to': 13, 'walk': 14, 'with': 15, 'world': 16}
```

```
[0.40546511 1.09861229 1.09861229 1.09861229 0.40546511 1.09861229
 1.09861229 0.40546511 1.09861229 1.09861229 1.09861229 1.09861229
 1.09861229 1.09861229 1.09861229 1.09861229 1.09861229]
```

NLP: TFIDF

- The term frequency inverse document frequency (TFIDF) combination simply scales the columns of the term frequency matrix by their inverse document frequency.

```
documents = ["Alex likes to walk with his dog",
             "Alex and his dog are best friends",
             "hello world this is random text"]
```

```
[[0.40546511 0.         0.         0.         0.40546511 0.
  0.         0.40546511 0.         1.09861229 0.         0.
  0.         1.09861229 1.09861229 1.09861229 0.        ]
 [0.40546511 1.09861229 1.09861229 1.09861229 0.40546511 1.09861229
  0.         0.40546511 0.         0.         0.         0.
  0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  1.09861229 0.         1.09861229 0.         1.09861229 1.09861229
  1.09861229 0.         0.         0.         1.09861229]]
```

NLP: Cosine Similarity

- Given a TFIDF (or just term frequency) matrix, one of the more common questions to address is to compute similarity between multiple documents in the corpus. The common metric for doing so is to compute the cosine similarity between two different documents.

$$CosineSimilarity(x, y) = \frac{x^T y}{\|x\|_2 \cdot \|y\|_2}$$

NLP: Cosine Similarity

```
documents = ["Alex likes to walk with his dog",
             "Alex and his dog are best friends",
             "hello world this is random text"]
```

```
[[1.         0.09269042 0.        ]
 [0.09269042 1.         0.        ]
 [0.         0.         1.        ]]
```

NLP: Word Embeddings

- Word embeddings are vector representations of words in a continuous vector space.

```
corpus = api.load('glove-wiki-gigaword-100')

vector = corpus['computer']
print(vector)
```

```
[-1.6298e-01  3.0141e-01  5.7978e-01  6.6548e-02  4.5835e-01 -1.5329e-01
  4.3258e-01 -8.9215e-01  5.7747e-01  3.6375e-01  5.6524e-01 -5.6281e-01
  3.5659e-01 -3.6096e-01 -9.9662e-02  5.2753e-01  3.8839e-01  9.6185e-01
  1.8841e-01  3.0741e-01 -8.7842e-01 -3.2442e-01  1.1202e+00  7.5126e-02
  4.2661e-01 -6.0651e-01 -1.3893e-01  4.7862e-02 -4.5158e-01  9.3723e-02
  1.7463e-01  1.0962e+00 -1.0044e+00  6.3889e-02  3.8002e-01  2.1109e-01
 -6.6247e-01 -4.0736e-01  8.9442e-01 -6.0974e-01 -1.8577e-01 -1.9913e-01
 -6.9226e-01 -3.1806e-01 -7.8565e-01  2.3831e-01  1.2992e-01  8.7721e-02
  4.3205e-01 -2.2662e-01  3.1549e-01 -3.1748e-01 -2.4632e-03  1.6615e-01
  4.2358e-01 -1.8087e+00 -3.6699e-01  2.3949e-01  2.5458e+00  3.6111e-01
  3.9486e-02  4.8607e-01 -3.6974e-01  5.7282e-02 -4.9317e-01  2.2765e-01
  7.9966e-01  2.1428e-01  6.9811e-01  1.1262e+00 -1.3526e-01  7.1972e-01
 -9.9605e-04 -2.6842e-01 -8.3038e-01  2.1780e-01  3.4355e-01  3.7731e-01
 -4.0251e-01  3.3124e-01  1.2576e+00 -2.7196e-01 -8.6093e-01  9.0053e-02
 -2.4876e+00  4.5200e-01  6.6945e-01 -5.4648e-01 -1.0324e-01 -1.6979e-01
  5.9437e-01  1.1280e+00  7.5755e-01 -5.9160e-02  1.5152e-01 -2.8388e-01
  4.9452e-01 -9.1703e-01  9.1289e-01 -3.0927e-01]
```

NLP: N-grams

- A sequence of n contiguous items (words or characters) in text.

```
n = 2
text = "Alex likes to walk with his dog, Alex and his dog are best friends"
```

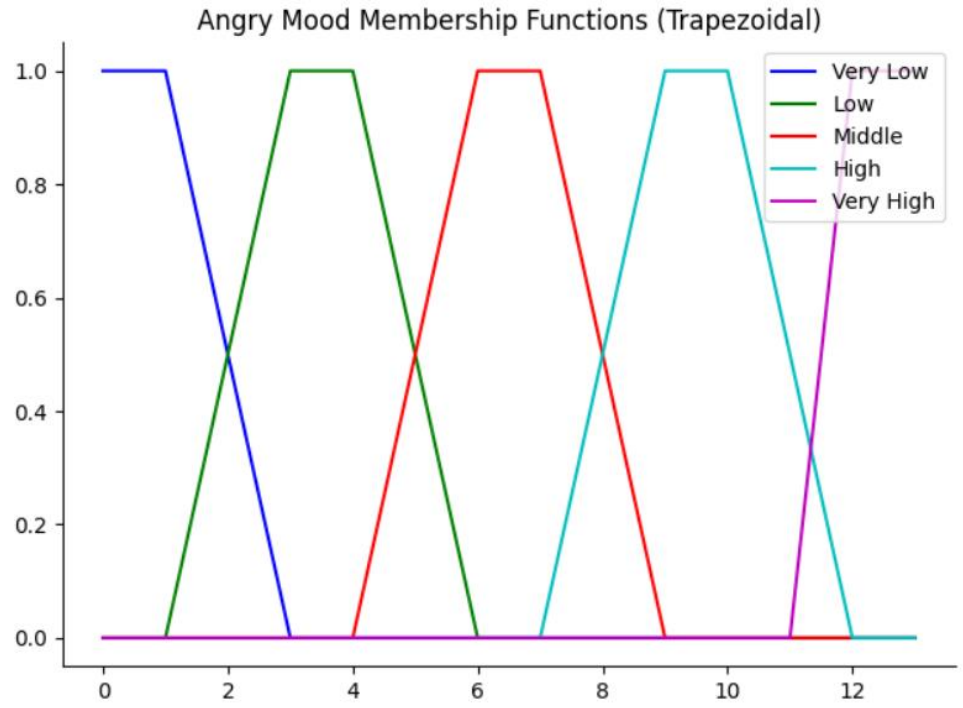
```
['alex likes', 'to walk', 'with his', 'dog alex', 'and his', 'dog are', 'best friends']
```

Fuzzy Logic

- Fuzzy logic is a form of logic that allows reasoning with uncertain or imprecise information.
- Unlike classical binary logic (true/false or 0/1), fuzzy logic works with degrees of truth.

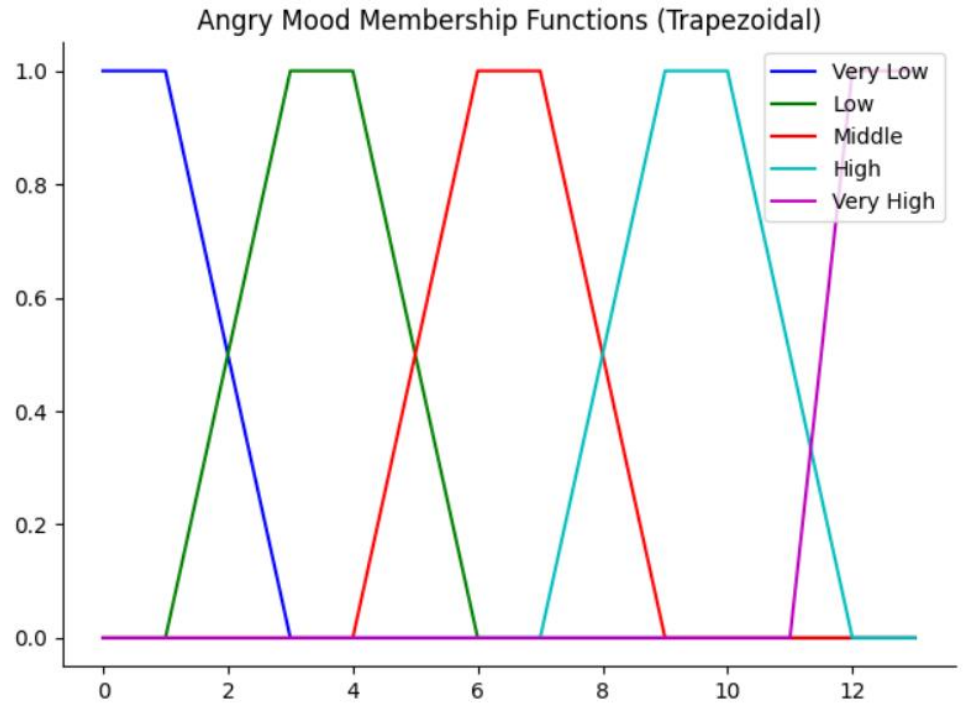
Fuzzy Logic: Membership Functions

- Membership functions define the degree to which a value belongs to a fuzzy set.
- Maps input values to a range [0, 1].
- 0 = completely not a member, 1 = fully a member, and values in between indicate partial membership.



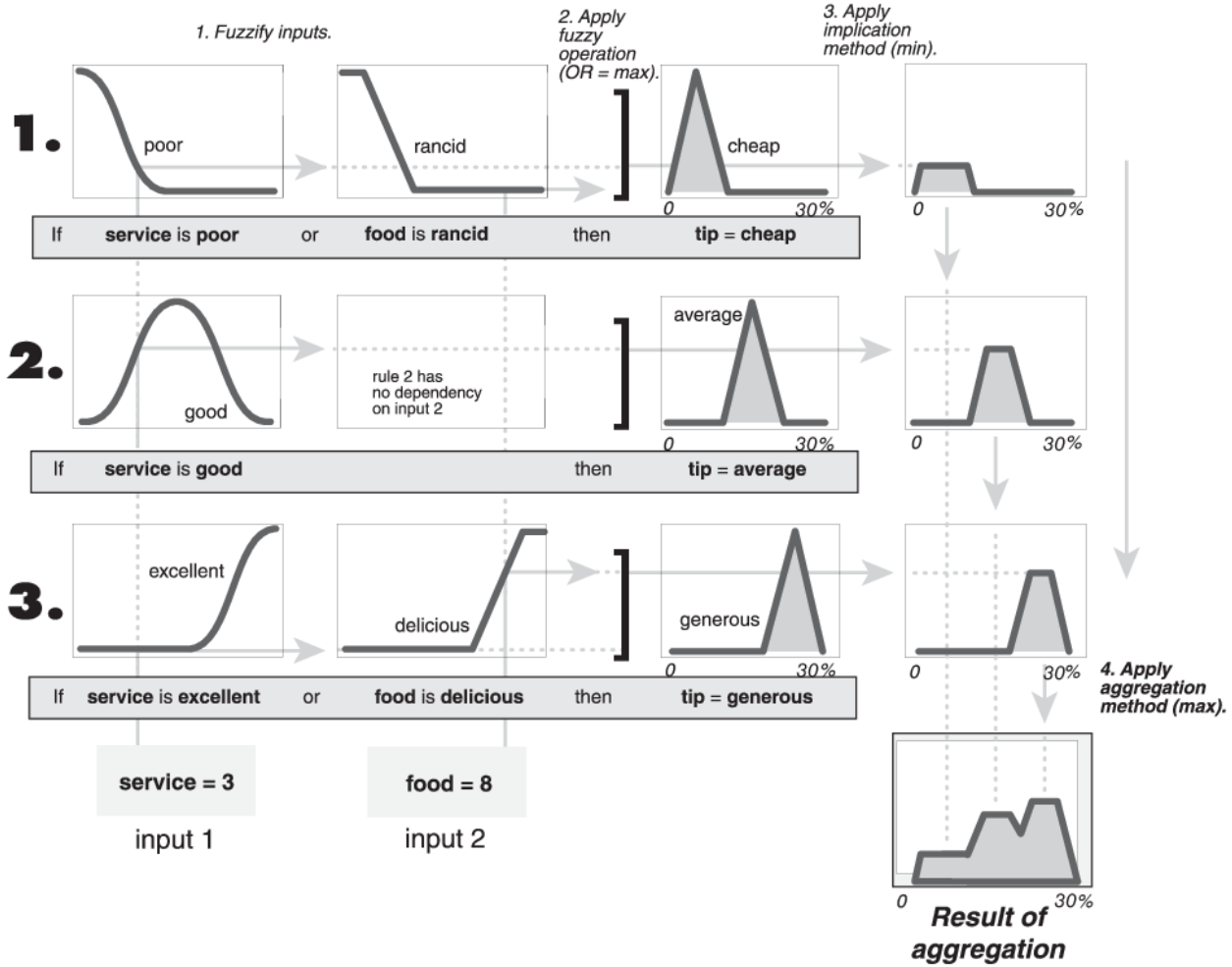
Fuzzy Logic: Fuzzy Rules

- Rules define how fuzzy sets interact to produce an output.
- Typically written as IF-THEN statements.
- IF temperature is high AND humidity is low THEN fan speed is high.



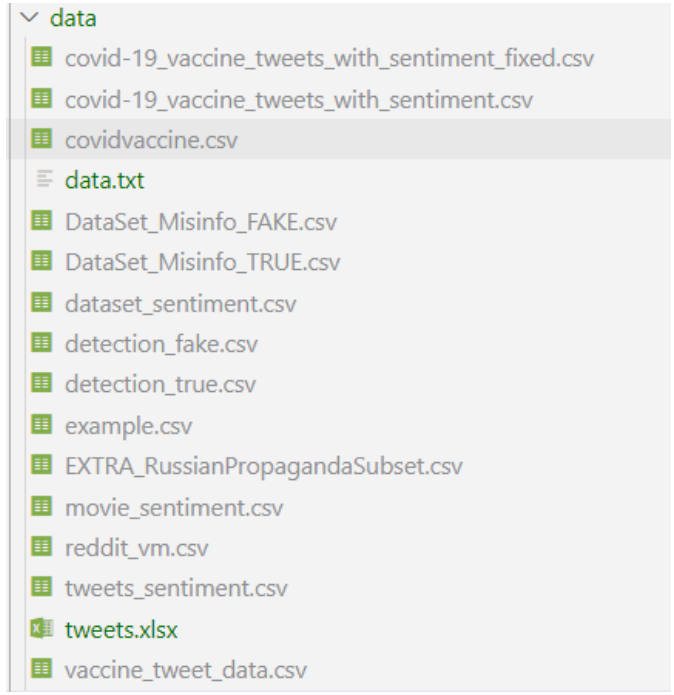
Fuzzy Logic: Inference Process

- Evaluate Rules: Determine the truth value of each rule's antecedent using membership functions.
- Combine Rules: Aggregate the results using operators like AND, OR, MIN, and MAX.
- Defuzzify: Convert the aggregated fuzzy output into a crisp value (e.g., using centroid or mean-of-max methods).



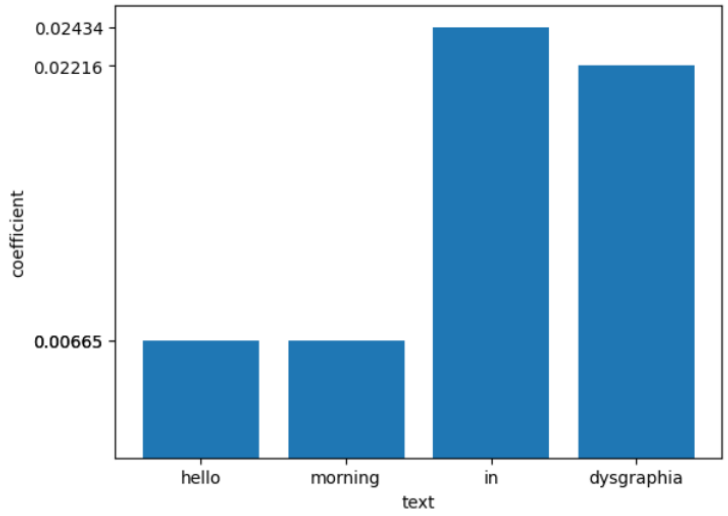
Disinformation Text Analysis

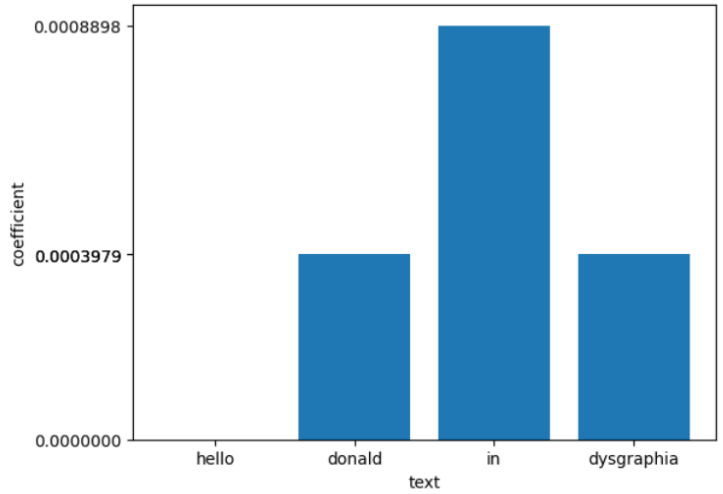
- This research explores the integration of Natural Language Processing (NLP) methods, specifically TF-IDF and n-gram analysis, with fuzzy logic rules employing Gaussian membership functions to detect disinformation in text data.
- The approach emphasizes reducing false positives by assessing the probability of disinformation rather than binary decisions, enhancing the accuracy and reliability of text analysis under informational uncertainty.



Disinformation Text Analysis

- In our research, we first used simple TF-IDF analysis to example texts. After applying n-gram approach, we saw a more confident detection for disinformation texts.





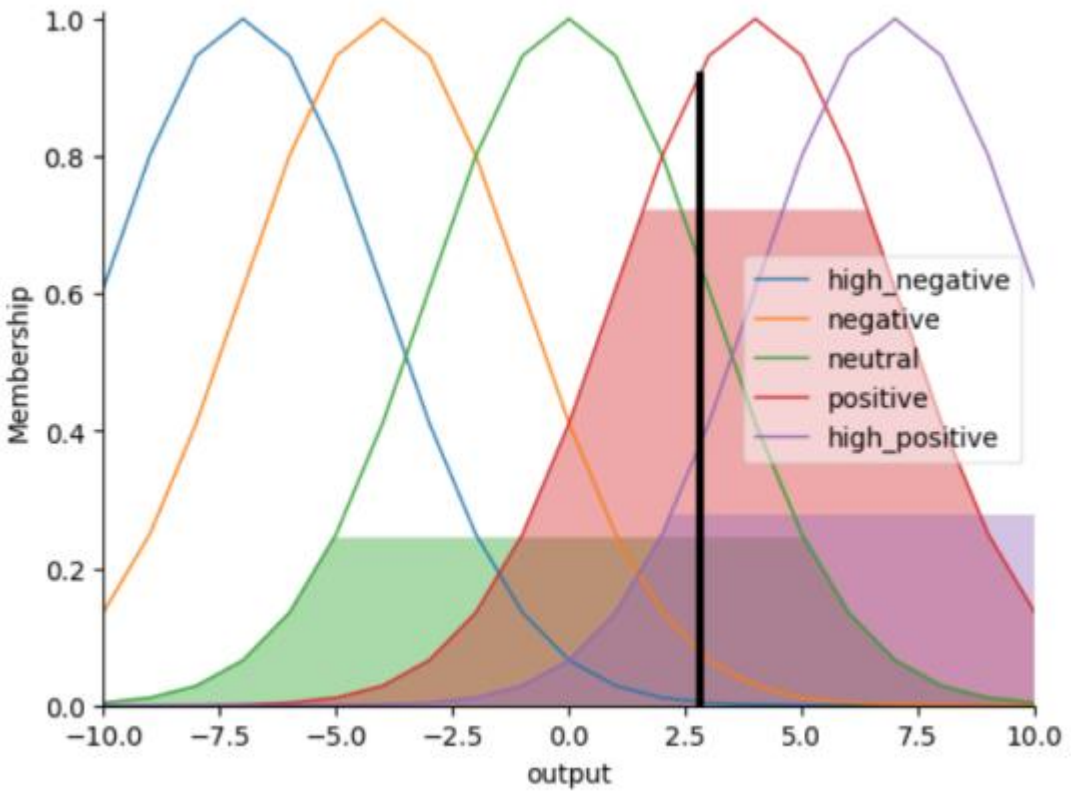
Disinformation Text Analysis

- We propose to use fuzzy logic rules to detect possible false-positives in TF-IDF and n-gram text analysis. Namely, we use another dataset with data samples, that can be misinterpreted as disinformation, but which are not actual examples of disinformation.

$c_{false-positive}$ \ $c_{disinfo}$	**low**	**middle**	**high**
low	neutral	high	very high
middle	low	neutral	high
high	very low	low	neutral

Disinformation Text Analysis

- By applying the fuzzy rules to our text example, we received a more confident detection. We also used Gaussian membership functions, as we do not require much calculation for two parameters c_false−positive and c_disinfo, and these membership functions provide more smoothness and concise notation as well as being nonzero at all points.



Text Sentiment Analysis

Understanding the emotional tone of textual data is crucial.

Traditional sentiment analysis techniques:

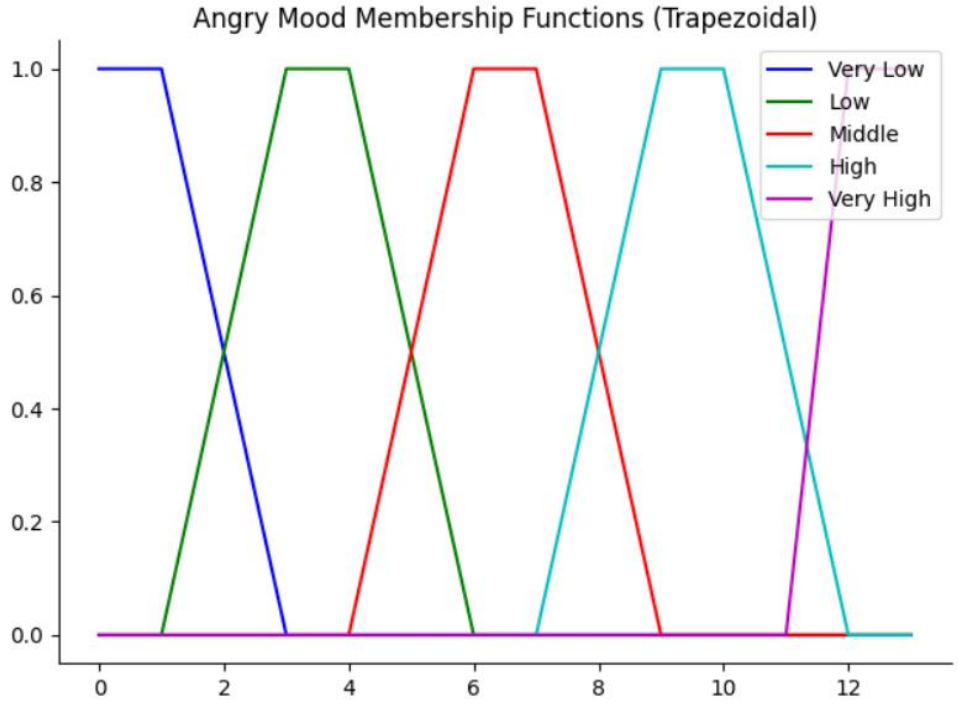
- often fall short because they rely on binary or categorical classifications that lack the granularity needed to capture the full spectrum of human emotions.

Using fuzzy logic to enhance the detection and interpretation of moods in text, providing a more nuanced and flexible approach.

Text Sentiment Analysis: Fuzzy Logic Approach

Fuzzy logic offers a robust framework for managing ambiguity and partial truths, making it an ideal tool for mood detection in text

- partial memberships across multiple emotional states
- single expression can belong to multiple categories to varying degrees



Text Sentiment Analysis: Fuzzy Logic Approach

- Defining linguistic variables for different emotional states ("Angry," "Sad," "Happy", etc.)
- Each variable is associated with fuzzy set characterized by trapezoidal and triangular membership functions.
- Construct a lexicon where words are mapped to these fuzzy sets, the fuzzification process transforms input text into a multidimensional representation of mood.

```
# Define the lexicon with mood vectors
moods = {0: "Angry", 1: "Worried", 2: "Sad", 3: "Calm", 4: "Happy", 5: "Excited"}
moods_indeces = {y.lower(): x for x, y in moods.items()}

lexicon = {
    "frustrated": [0.8, 0.4, 0.3, 0, 0, 0], "anxious": [0.3, 0.9, 0.4, 0, 0, 0.1],
    "disappointed": [0.5, 0.3, 0.8, 0.1, 0, 0], "furious": [0.9, 0.3, 0.2, 0, 0, 0],
    "peaceful": [0, 0, 0, 1, 0.4, 0.2], "hate": [0.9, 0.3, 0.3, 0, 0, 0],
    "joyful": [0, 0, 0, 0.2, 0.9, 0.7], "unacceptable": [0.9, 0.6, 0.6, 0, 0, 0],
    "thrilled": [0, 0, 0, 0, 0.6, 1], "infuriating": [0.9, 0.8, 0.6, 0, 0, 0],
    "irate": [0.9, 0.2, 0.1, 0, 0, 0], "terrible": [0.9, 0.7, 0.7, 0, 0, 0],
```

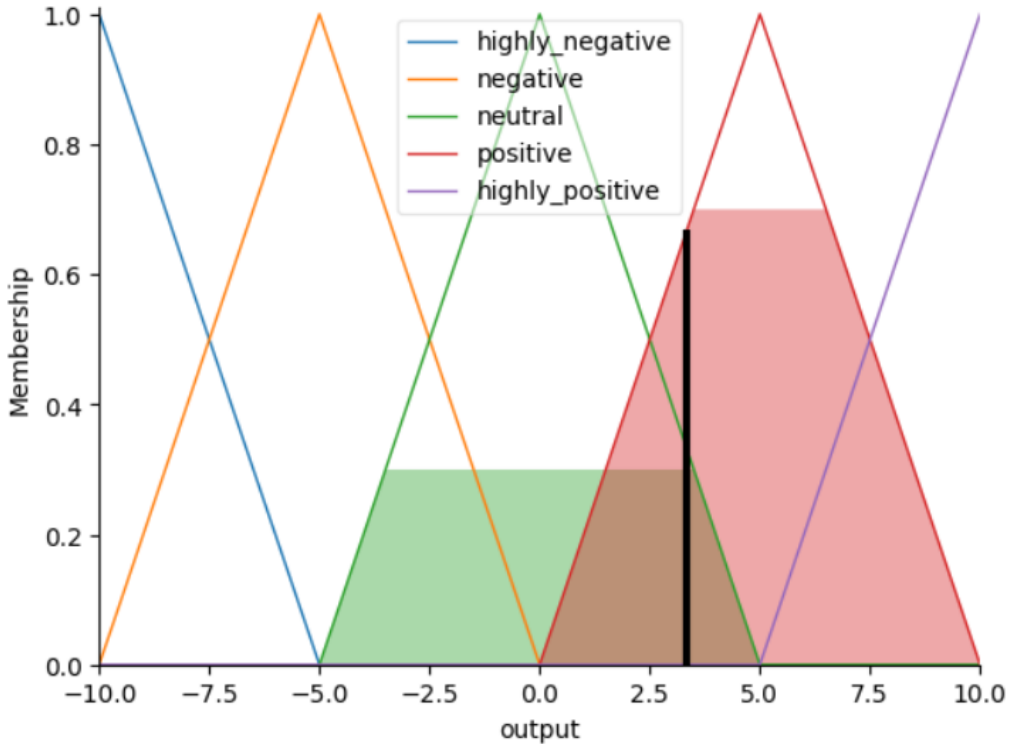
Text Sentiment Analysis: Fuzzy Inference System

- Fuzzy inference system: a set of rules determines how different emotional indicators combine to produce an overall mood prediction.
- Intensity and combination of emotions, a nuanced interpretation of the text's emotional tone.
- Example: high values for both "Happy" and "Calm," the overall mood is likely positive.

```
rules_tensor = [
    'angry high or happy low then negative',
    'angry low and happy high then positive',
    'angry medium or happy medium then neutral',
    'angry high and worried high then high_negative',
    'angry medium and worried medium then negative',
    'angry low and worried low then neutral',
```

Text Sentiment Analysis: Results

- Effectiveness of the fuzzy logic-based approach in improving mood detection accuracy.
- Accuracy, precision, recall, and F1 score.
- Capturing more nuanced emotional expressions, reducing false positives, and providing a more reliable assessment of mood dynamics.



Text Sentiment Analysis: Discussion

Pros:

- flexibility and adaptability to the ambiguous nature of human language
- accommodating degrees of membership across multiple emotional categories, provides a more accurate representation of mood intensities and combinations

Cons:

- need for extensive domain knowledge to develop the lexicon and inference rules
- scalability and adaptability to new domains remain areas for future improvement

Future Research Directions

- Expansion and refinement of the lexicon using machine learning techniques
- Integrating context-aware analysis, which could significantly improve the accuracy of mood detection by accounting for the broader discourse
- Real-time processing capabilities to support dynamic NLP applications in interactive environments

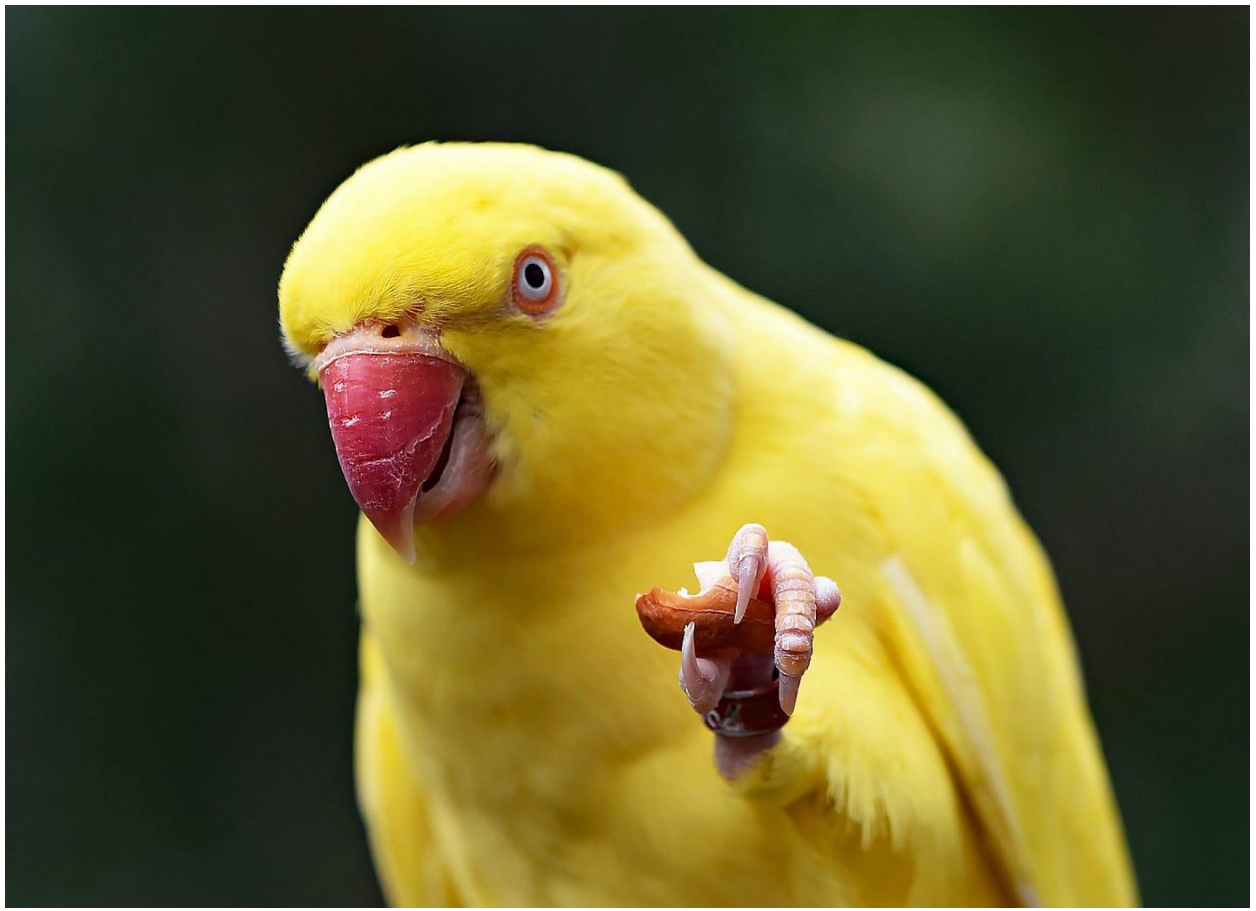
Conclusion

- Fuzzy logic provides a powerful tool for enhancing disinformation detection and sentiment analysis in NLP
- Fuzzy logic models can better capture the nuanced spectrum of human emotions compared to traditional methods

References

- [1] L. A. Zadeh, “Fuzzy sets,” Information and control, vol. 8 (1965), pp. 338–353.
- [2] L. A. Zadeh, “Similarity relations and fuzzy orderings,” Information sciences, vol. 3 (1971), pp. 177–200.
- [3] B. Pang, L. Lee, and Sh. Vaithyanathan, “Thumbs up? Sentiment Classification using Machine Learning Techniques,” Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002), pp. 79–86.
- [4] S. Mohammad, and P. Turney, “Crowdsourcing a word-emotion association lexicon,” Computational Linguistics, 2013, URL: https://arxiv.org/abs/1308.6297
- [5] X. Wang, H. Zhang, and Zh. Xu, “Public Sentiments Analysis Based on Fuzzy Logic for Text,” International Journal of Software Engineering and Knowledge Engineering, 2016, 26:09n10, pp. 1341-1360.

Q&A



Thank you!