Hierarchical Attention Networks for Document Classification

Zichao Yang, Diyi Yang, Chris Dyer, Xiadong He,
Alex Smola, Eduard Hovy

Presented by Pamudu Ranasinghe

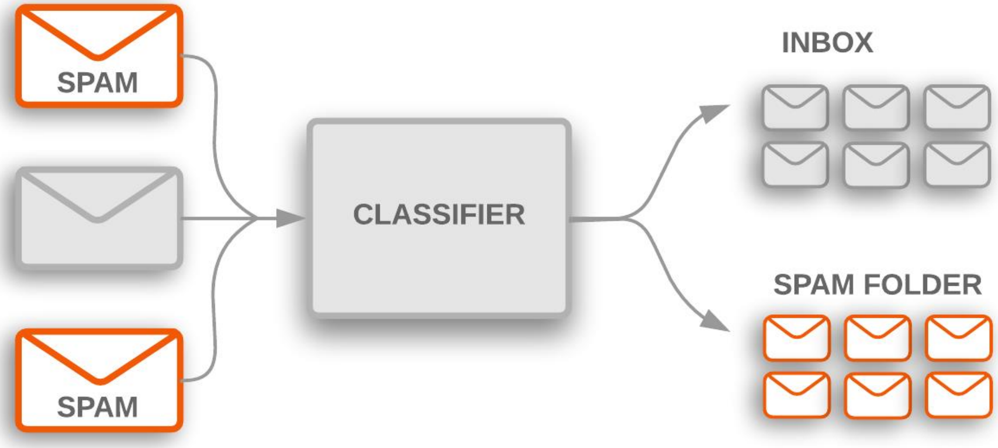
Text Classification

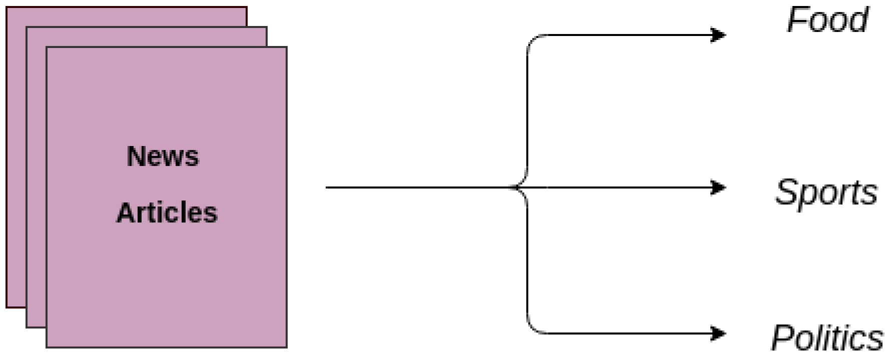
The goal is to automatically assign predefined labels or categories to a piece of text.

- 🔖 **Topic Labelling**
  Is this article about 'Food', 'Sports', or 'Politics'?

- 😊 **Sentiment Classification**
  Is this product review 'Positive', 'Negative', or 'Neutral'?

- 📧 **Spam Detection**
  Is this email 'Spam' or 'Not Spam'?



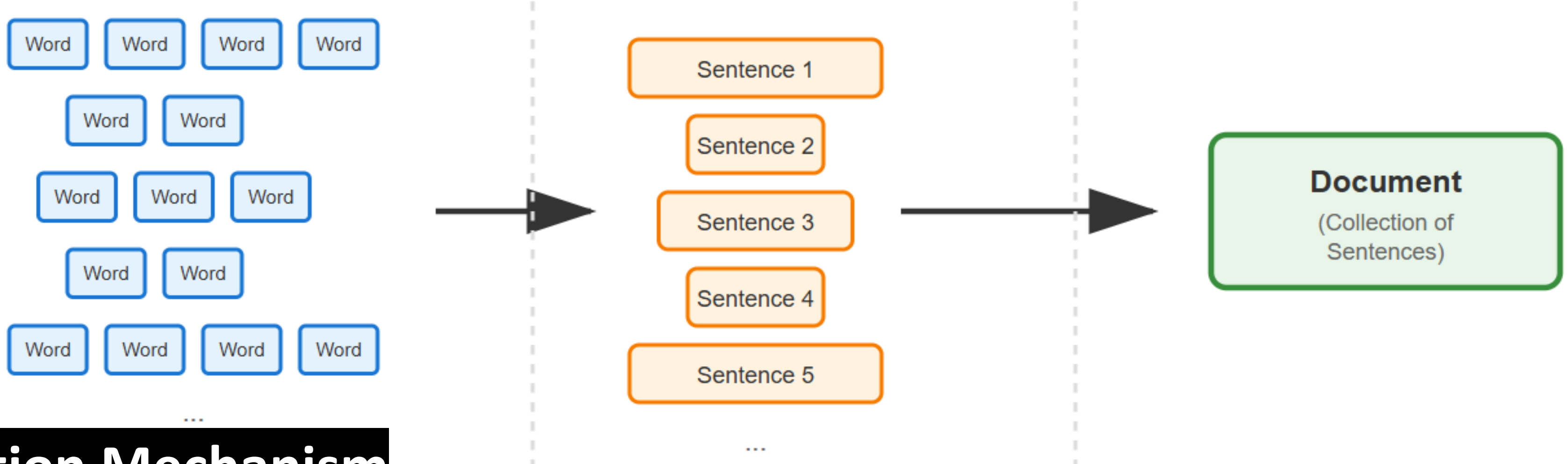


Existing Methods [Text Classification]

Linear & SVM Models	Standard Neural Networks	Hierarchical Models
Bag-of-Words/N-grams: Uses frequency counts of the most common words and phrases. **Bag-of-Means:** Averages the pre-trained embeddings (word2vec) of all words in a document. **SVM + Text Features:** An SVM model using features like n-grams and sentiment lexicons.	**Word/Char CNN:** A Convolutional Neural Network learns features from word or character sequences for classification. **LSTM:** Reads an entire document as a sequence and averages the hidden states of all words for classification.	**Conv-GRNN:** A hierarchical model that uses a CNN to create sentence vectors, which are then combined by a Gated Recurrent Neural Network (GRNN). **LSTM-GRNN:** Similar to Conv-GRNN, but it uses an LSTM to create the sentence vectors.

The Intuition Behind HAN [Hierarchical Attention Network]

Hierarchical Structure Documents have a natural hierarchy: words form sentences, and sentences form a document.



The Attention Mechanism

Not all parts of a document are equally important for understanding its meaning.

The Solution: The model should learn to pay more attention to the words and sentences that are most informative.

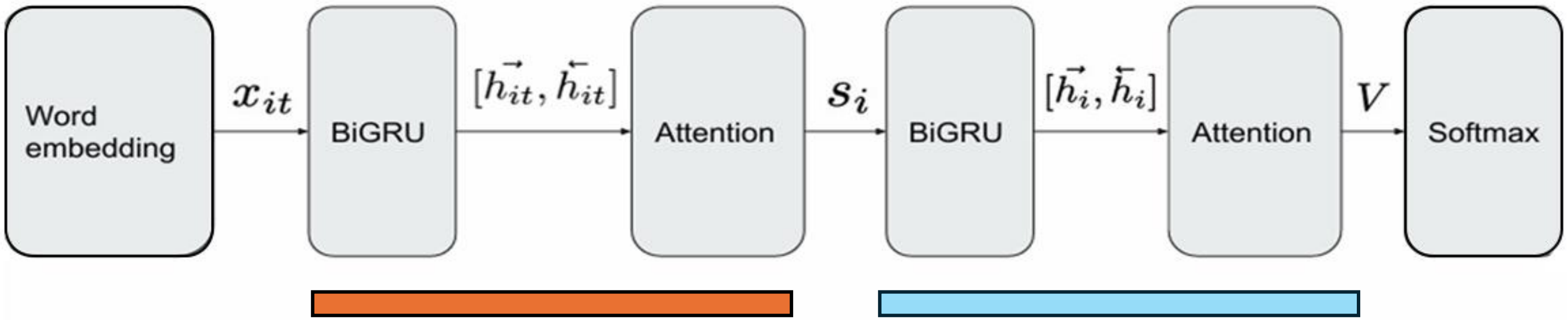
The waiter was friendly.

But the wait time for our food was **long**.

When the food arrived, it was **perfect**.

I might come back.

Model Architecture



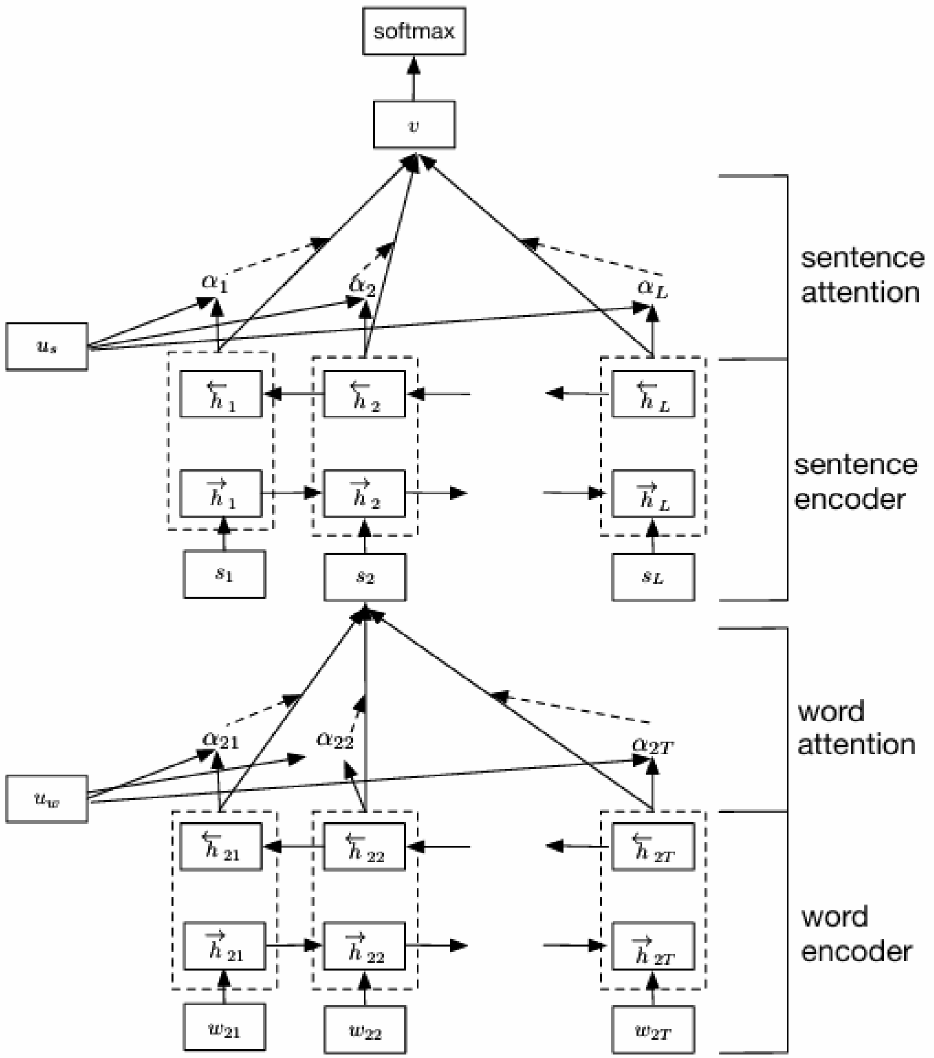
Words to Sentences

- **Word Encoder (BiGRU):** Captures the meaning of each word based on its context within the sentence.
- **Word Attention:** Identifies and weighs the most informative words to create a consolidated sentence.

Sentences to Document

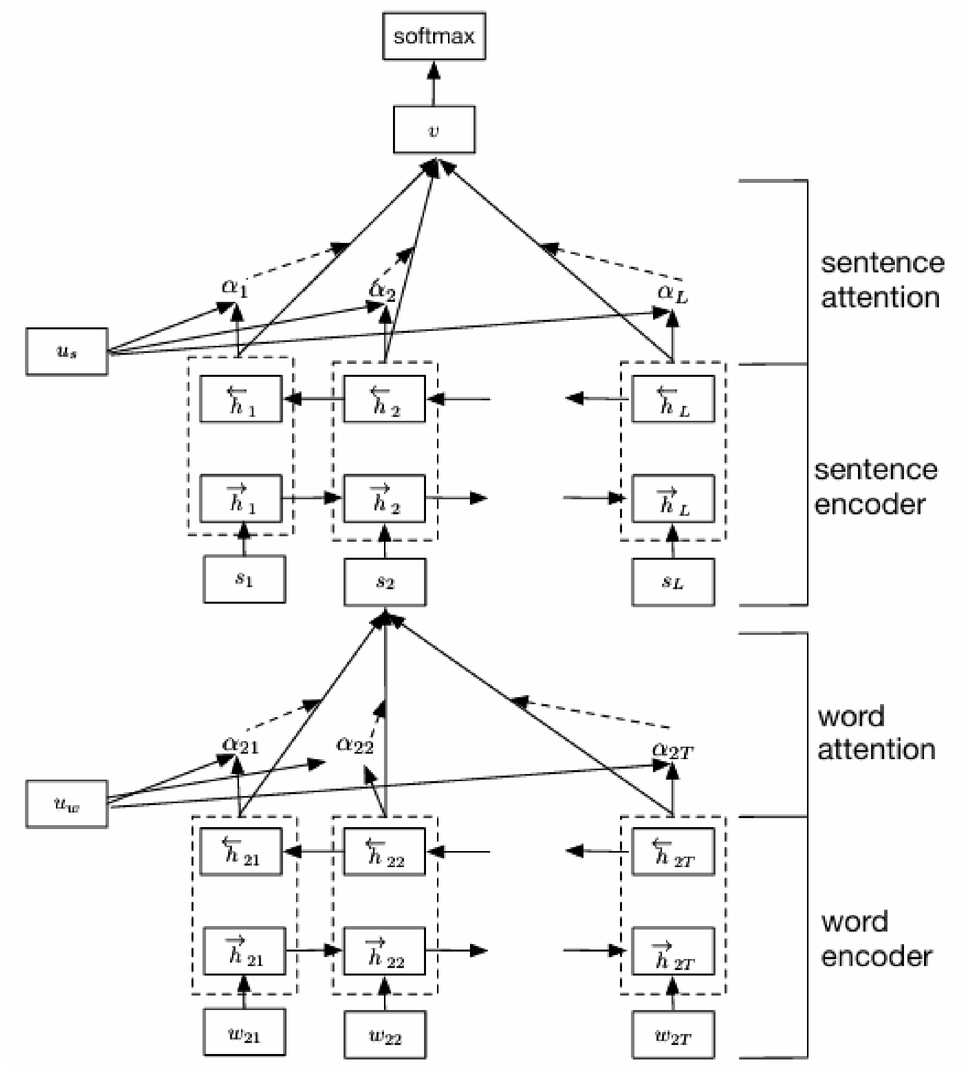
- **Sentence Encoder (BiGRU):** Captures the meaning of each sentence based on the context of surrounding sentences.
- **Sentence Attention:** Identifies and weighs the most informative sentences to create a final document.

Model Architecture [2]




```
function build_sentence_vector(sentence):
    # Word-level BiGRU
    word_annotations = WordEncoder(sentence) # Returns h_it for all words

    # Word-level Attention
    sentence_vector = WordAttention(word_annotations) # Returns s_i
    return sentence_vector


function build_document_vector(sentence_vectors):
    # Sentence-level BiGRU
    sentence_annotations = SentenceEncoder(sentence_vectors) # Returns h_i

    # Sentence-level Attention
    document_vector = SentenceAttention(sentence_annotations) # Returns v
    return document_vector


function HierarchicalAttentionNetwork(document):
    # 1. Break document into sentences
    sentences = split_into_sentences(document)

    # 2. Get a vector for each sentence
    for sentence in sentences:
        sentence_vector = build_sentence_vector(sentence)
        sentence_vectors.append(sentence_vector)

    # 3. Build the final document vector
    document_vector = build_document_vector(sentence_vectors)

    # 4. Classify the document
    probabilities = softmax(document_vector)
    return probabilities
```

Model Architecture [3]




```
# Encoder Function (using a Bidirectional GRU)
# Input: A sequence of vectors (e.g., word embeddings or sentence vectors)
# Output: A sequence of context-aware annotation vectors
function Encoder(input_sequence):
    # 1. Process the sequence from start to end
    forward_hidden_states = RNN_forward(input_sequence)

    # 2. Process the sequence from end to start
    backward_hidden_states = RNN_backward(input_sequence)

    # 3. Concatenate the states for each item in the sequence
    annotations = concatenate(forward_hidden_states, backward_hidden_states)

    return annotations


# Attention Function
# Input: A sequence of annotation vectors from an Encoder
# Output: A single, context-aware vector representing the entire sequence
function Attention(annotations):
    # Parameters learned during training for this attention level
    # W_attention: weight matrix
    # b_attention: bias vector
    # u_context: context vector

    # 1. Create a hidden representation of the annotations
    hidden_representation = tanh(annotations * W_attention + b_attention)

    # 2. Calculate importance scores by comparing with the context vector
    attention_scores = dot_product(hidden_representation, u_context)

    # 3. Normalize scores to get final attention weights
    attention_weights = softmax(attention_scores)

    # 4. Create the final vector as a weighted sum of annotations
    final_vector = sum(attention_weights * annotations)

    return final_vector
```

HN - Model Comparison

	HN-ATT (Attention	HN-AVE (Averaging)	HN-MAX (Max-Pooling)
Mechanism	Uses a learned, weighted sum based on context vectors.	Performs a simple average of word/sentence annotations.	Selects the feature with the maximum value over time from the word/sentence annotations.
Key Implication	The superiority of this model clearly demonstrates the effectiveness of learning context-dependent word and sentence importance.	**This model is equivalent to using non-informative context vectors (uniform attention).** Its solid baseline performance shows the benefit of hierarchy alone.	The fact that this method does not beat simple averaging suggests that **capturing a single, most salient feature is less effective** than aggregating information, even uniformly.

Accuracy

Datasets

The model was evaluated across **06 large-scale datasets** for **02 key NLP tasks**.

😳 Sentiment Classification

- **Yelp Reviews (2013-2015)**
- **IMDB Reviews**
- **Amazon Reviews**

🔖 Topic Classification

- **Yahoo Answers**

Data set	classes	documents	average #s	max #s	average #w	max #w	vocabulary
Yelp 2013	5	335,018	8.9	151	151.6	1184	211,245
Yelp 2014	5	1,125,457	9.2	151	156.9	1199	476,191
Yelp 2015	5	1,569,264	9.0	151	151.9	1199	612,636
IMDB review	10	348,415	14.0	148	325.6	2802	115,831
Yahoo Answer	10	1,450,000	6.4	515	108.4	4002	1,554,607
Amazon review	5	3,650,000	4.9	99	91.9	596	1,919,336

Model configuration and training

Data & Embeddings

- **Vocabulary:** Words appearing less than 5 times were replaced with a special **UNK token**.
- **Word Vectors:** Initialized with pre-trained **200-dimensional word2vec embeddings.**

Model Specifications

- **GRU Size:** 50 hidden units for each forward/backward GRU.
- **Annotation & Context Vectors:** 100 dimensions (combining forward & backward GRUs).

Training Process

- **Optimizer & Batching:**
  Stochastic Gradient Descent (SGD) with a batch size of 64.
  Documents of similar length were batched together, which accelerated training by 3x.
- **Hyperparameters:** The best learning rate was found using grid search on the validation set.

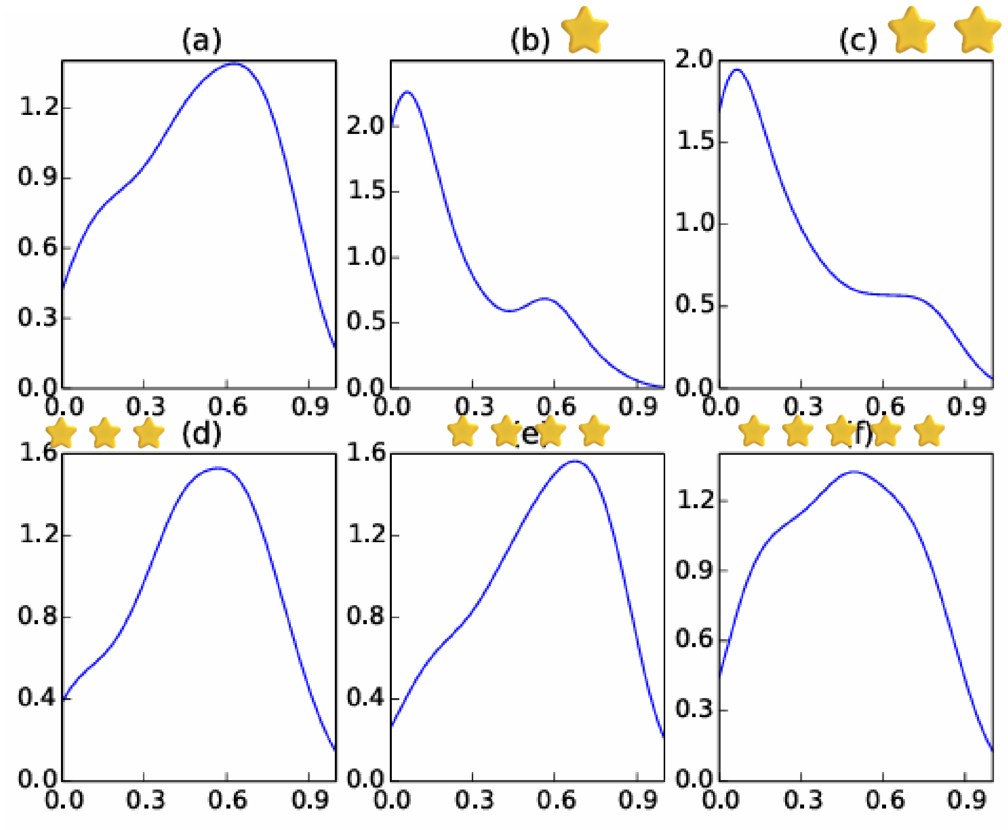
Performance Comparison

	Yelp'13	Yelp'14	Yelp'15	IMDB	Yahoo
BoW	-	-	58.0	-	68.9
BoW TFIDF	-	-	59.9	-	71.0
Majority	35.6	36.1	36.9	17.9	-
SVM + Bigrams	58.9	60.0	61.1	39.9	-
SVM + Unigrams	57.6	61.6	62.4	40.9	-
LSTM	-	-	58.2	-	70.8
CNN-char	-	-	62.0	-	71.2
Conv-GRNN	63.7	65.5	66.0	42.5	-
LSTM-GRNN	65.1	67.1	67.6	45.3	-
HN-AVE	67.0	69.3	69.9	47.8	75.2
HN-MAX	66.9	69.3	70.1	48.2	75.2
HN-ATT	68.2	70.5	71.0	49.4	75.8

Legend

- Best (≥70%)
- Good (60-69%)
- Average (50-59%)
- Low (<50%)
- No Data
- HN Methods
- Previous Best

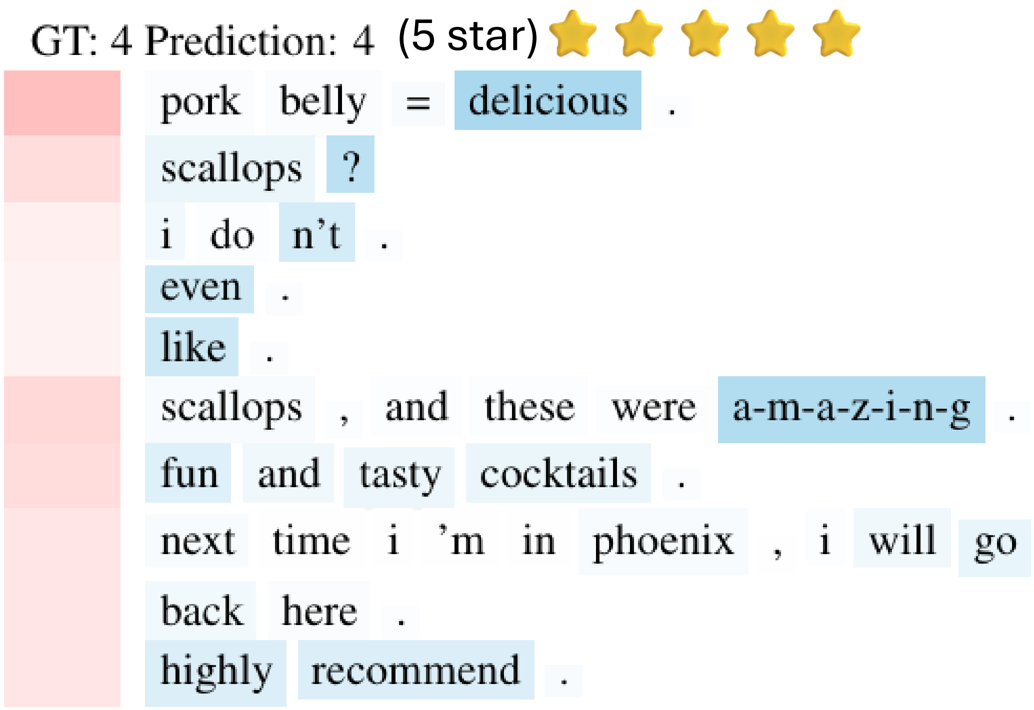
Results and Analysis

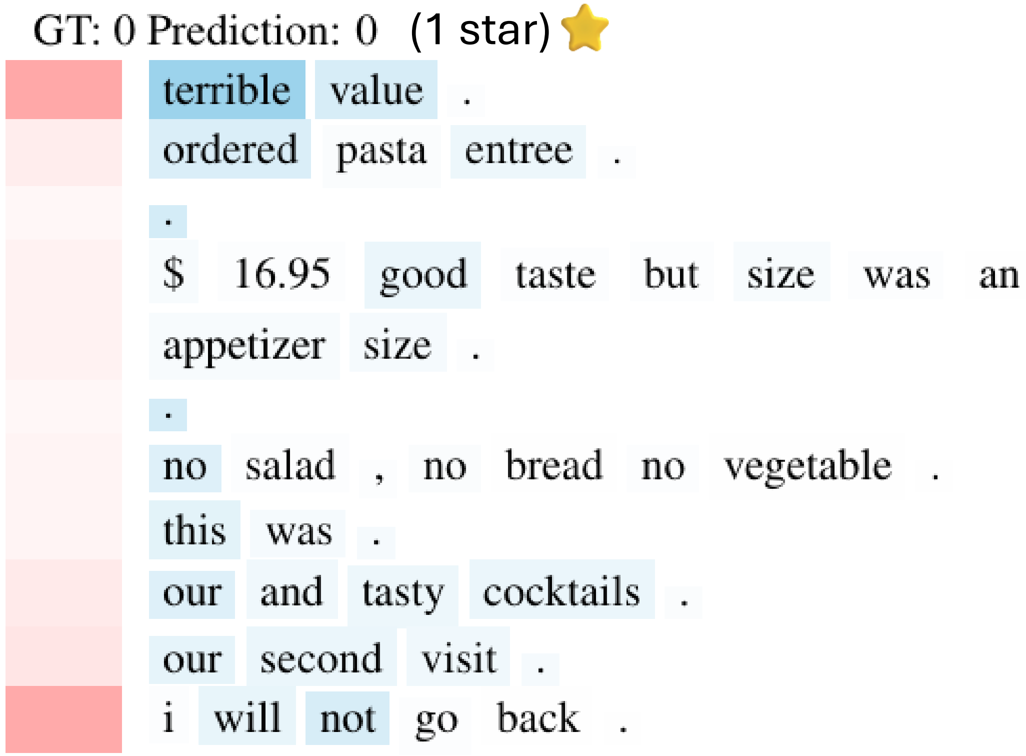


(a) Aggregate Distribution: The overall attention given to the word "good" across all reviews.
(b) 1-Star Reviews: The model learns to ignore "good" in negative reviews, assigning it a very low weight.
(c) 2-Star Reviews: Attention remains low, as "good" is still not a reliable positive indicator.
(d) 3-Star Reviews: The distribution is more centered, reflecting mixed or neutral sentiment.
(e) 4-Star Reviews: The model now pays high attention, recognizing "good" as an important positive word.
(f) 5-Star Reviews: "good" consistently receives high attention as a key indicator of a very positive review.

- **What this shows:** The distribution of attention weights assigned to the word "**good**" based on the review's star rating.
- **X-axis:** Attention Weight (Low to High Importance).
- **Y-axis:** Frequency.
- **The Trend:** As the star rating increases, the model pays **more** attention to the word "good".

Results and Analysis [2]





- **Red Highlight:** Indicates the importance of the **sentence**. A darker red means the model found the entire sentence more important.
- **Blue Highlight:** Indicates the importance of a **word** within its sentence. A darker blue means the word was more influential for that sentence's meaning.

Practical Considerations & Challenges

🔁 Costly to Add New Categories

The model is trained for a fixed set of classes.

Adding a new class (e.g., a new topic or sentiment) requires modifying the final layer and fully retraining the model—making it **inefficient for agile workflows** with changing requirements.

⚙️ Computationally Intensive

Bidirectional GRUs + Attention = Heavy computation.

Training demands powerful GPUs and long durations due to the model's complexity.

📊 Requires Large Datasets

Performance scales with data.

HAN performs best with large datasets. On smaller datasets, it may underperform and become prone to overfitting compared to simpler models.

Thank You