Medical Abbreviation Disambiguation Using Naive Bayes and Neural Networks

Vihaan Manchanda
vm180@duke.edu

Anvita Suresh
as1693@duke.edu

Arushi Singh
as1685@duke.edu

Duke University
Introduction to Natural Language Processing
Instructor: Patrick Wang
November 19, 2025

1 Motivation

Medical abbreviations create significant safety risks in healthcare. A single abbreviation like "CP" can represent Chronic Pain, Chest Pain, or Cerebral Palsy, each requiring dramatically different treatments. The Joint Commission reports that abbreviation-related errors consistently rank among the top causes of sentinel events [2].

Consider this scenario: A physician writes "Patient presents with CP and difficulty breathing." Without proper context, this could be interpreted as Chest Pain (requiring immediate cardiac workup for potential myocardial infarction) when the physician actually meant Chronic Pain with anxiety-related hyperventilation. Conversely, misinterpreting Chest Pain as Chronic Pain could delay critical cardiac intervention. If the context involves a pediatric patient and CP refers to Cerebral Palsy, the treatment approach differs entirely, focusing on motor function and developmental support rather than acute symptom management.

The problem extends beyond patient safety. Clinical decision support systems, automated coding, and health information exchange platforms rely on accurate text interpretation. Ambiguous abbreviations force manual review, creating workflow bottlenecks.

This project addresses abbreviation disambiguation as classification: Given an ambiguous abbreviation and surrounding context, can we automatically predict the correct expansion? Success would enable safer documentation, more reliable automated systems, and reduced cognitive load on healthcare providers.

2 Related Work

2.1 Inspiration: The MeDAL Paper

Our work is inspired by Wen et al. (2020) [1], who created MeDAL (Medical Dataset for Abbreviation Disambiguation). Their goal differed from ours: they used abbreviation disambiguation as a pretraining task to improve performance on downstream medical NLP tasks like mortality and diagnosis prediction. They employed LSTM, LSTM with Self-Attention, and ELECTRA transformers, achieving 82–84% accuracy.

2.2 How Our Approach Differs

We diverge in three ways. First, we treat disambiguation as the primary problem, focusing on interpretability, computational efficiency, and understanding failure modes rather than transfer learning. Second, we compare classical probabilistic methods (Multinomial Naive Bayes) with neural network approaches (feedforward networks with BioWordVec embeddings and attention mechanisms) rather than using complex sequential models like LSTMs or transformers. Third, we evaluate on a carefully filtered subset with clear acronym patterns (first letters match), whereas MeDAL used the full dataset.

Our question: Can simpler classical and neural methods compete with sophisticated deep learning architectures when given appropriate features? What can the performance gap between bag-of-words and embedding-based approaches reveal about medical abbreviation disambiguation's fundamental difficulty?

3 Implementation Overview

3.1 Problem Formulation

We formulate disambiguation as multi-class classification. Given abbreviation a and context words $w_1, ..., w_n$, predict expansion e^*:

$$e^* = \arg\max_{e \in E} P(e|w_1, ..., w_n)$$

For our three selected abbreviations, we have:

- CC: {colorectal cancer, cell culture, cervical cancer}
- CP: {chronic pain, chest pain, cerebral palsy}
- SA: {surface area, sleep apnea, substance abuse}

This creates a 9-class classification problem.

3.2 Why Multinomial Naive Bayes?

We selected Naive Bayes for five specific reasons:

1. Bag-of-words matches medical terminology. Distinctive keywords signal meanings: "colon" suggests colorectal cancer, "flask" suggests cell culture. Naive Bayes captures these associations without requiring sequential information.

2. Independence assumption is reasonable. Within a 5-word window, most words are not grammatically dependent. "Colon" three words before and "tumor" two words after independently provide evidence for colorectal cancer.

3. Interpretability. Explicit $P(w|e)$ probabilities enable straightforward failure analysis. We can examine which keywords drive predictions.

4. Data efficiency. Performs well with limited training data compared to deep learning requiring millions of examples.

5. Computational efficiency. Training is $O(nd)$, inference is $O(kd)$. Practical for real-time clinical decision support.

3.3 Why Test TF-IDF?

We hypothesized TF-IDF would improve accuracy by down-weighting common medical stopwords ("the", "of", "patients") that dominate feature space but provide little discriminative power. TF-IDF: TF-IDF(w, d) = TF$(w, d) \times \log \frac{N}{\text{DF}(w)}$ reduces influence of high document frequency words while preserving context-specific terms. We test this in Section 7.

3.4 Why Neural Networks with BioWordVec?

While Naive Bayes provides interpretability and computational efficiency, neural networks offer the potential to capture non-linear relationships between context words and abbreviation meanings. We implement two neural network architectures to test whether learned representations outperform explicit probabilistic models:

1. Semantic representation through embeddings. Pre-trained BioWordVec embeddings [3] encode medical domain knowledge in dense 200-dimensional vectors. Unlike bag-of-words features that treat "colon" and "bowel" as unrelated, BioWordVec places semantically similar medical terms close together in embedding space. This addresses Naive Bayes's limitation of semantic similarity blindness.

2. Non-linear feature combinations. Neural networks with ReLU activations can learn non-linear decision boundaries. For example, the combination of "screening" + "women" might indicate cervical cancer more strongly than either word alone. Naive Bayes assumes conditional independence and cannot capture such interactions.

3. Learned feature weighting. Rather than treating all context words equally, neural networks can learn which words are most discriminative. Our attention mechanism explicitly models this by assigning learned importance weights to each context word.

4. Addressing context window limitations. While both Naive Bayes and neural networks use the same ± 5 word window, neural networks can learn distributed representations that capture semantic relationships even when discriminative keywords fall outside the window.

5. Comparison to classical methods. Testing neural networks allows us to quantify the performance gap between classical and modern approaches. If the gap is small, this validates simpler models for clinical deployment. If the gap is large, this justifies the additional complexity.

Why BioWordVec specifically? We selected BioWordVec over general-purpose embeddings (Word2Vec, GloVe) for three reasons: (1) training corpus includes PubMed abstracts and MIMIC-III clinical notes, ensuring coverage of medical terminology; (2) proven effectiveness on biomedical NLP tasks [3]; (3) publicly available pre-trained 200-dimensional vectors, eliminating training overhead.

4 Data Selection and Filtering

4.1 Dataset Choice

We used MeDAL (14.4 million PubMed abstracts with automatically labeled abbreviations via reverse substitution). We chose MeDAL over our original proposal (MIMIC-III) for three reasons: (1) labels already exist, (2) public availability without credentialing, (3) scale and diversity across medical specialties.

4.2 Filtering Pipeline

The full dataset contains 5,886 abbreviations. Our four-stage filtering:

Stage 1: Multi-word filter. Keep only multi-word expansions (e.g., "colorectal cancer" not "cancer") to ensure true acronyms.

Stage 2: First-letter matching. Verify first letters match abbreviation. Removes spurious labels like "PT" → "tumors".

Stage 3: Minimum sense filter. Require ≥3 expansions per abbreviation for meaningful classification.

Stage 4: Balance and relevance. Select abbreviations with 50K+ examples, reasonable balance (no class >40%), and pure medical terminology.

The filtering script (`preprocessing/filter_data.py`) processes 14.3 million examples, producing 113,371 filtered examples (0.79% of original).

4.3 Final Selection

Abbrev	Top 3 Expansions	Count	%
CC	colorectal cancer	28,201	30.4%
	cell culture	18,441	19.9%
	cervical cancer	14,876	16.0%
CP	chronic pain	10,183	18.3%
	chest pain	7,953	14.3%
	cerebral palsy	5,258	9.5%
SA	surface area	15,936	27.2%
	sleep apnea	6,903	11.8%
	substance abuse	5,620	9.6%

Table 1: Selected abbreviations (total: 113,371 examples).

Why these three? **CC** tests organ-specific cancer distinction plus laboratory context. **CP** tests temporal vs. anatomical framing with neurological dimension. **SA** spans measurement, respiratory, and behavioral domains, testing cross-specialty disambiguation.

4.4 Synthetic Data Generation

Step 3a requires synthetic data aligning with Naive Bayes assumptions: independent words with distinctive keywords per class. This tests whether our model works under ideal conditions. High accuracy on synthetic but low on real data indicates real-world assumption violations.

We used template-based generation with keyword slots:

```
"Patient with {abbrev} showing {kw1} and {kw2} findings"
```

We manually defined discriminative keywords based on domain knowledge. Examples: colorectal cancer (colon, rectal, tumor, polyp), cell culture (medium, serum, flask, cells), cerebral palsy (motor, spastic, children, pediatric), sleep apnea (obstructive, CPAP, snoring, apneic).

For each expansion, we generated 200 unique examples (1,800 total, perfectly balanced) by randomly selecting templates and keywords. Why templates rather than sampling real data? Sampling would create synthetic data too similar to our test set. Templates create idealized examples where keywords perfectly discriminate, representing best-case scenarios.

The generation script (`preprocessing/generate_synthetic.py`) ensures uniqueness by rejecting duplicates.

5 Feature Extraction and Model Implementation

5.1 Feature Representation in Naive Bayes

We represent each example as a bag-of-n-grams within a fixed context window:

Algorithm 1 Feature Extraction

Input: Text, abbreviation location
Output: Feature vector

Extract words within ±5 positions of abbreviation
Generate unigrams from context words
Generate bigrams from consecutive word pairs
Generate trigrams from consecutive word triples
Convert to count vector using vocabulary
Return: sparse count vector

Why ±5 word window? We tested different window sizes in preliminary experiments. Windows smaller than 5 often missed discriminative keywords. Windows larger than 7 introduced noise and increased computational cost without improving accuracy.

Why n-grams up to 3? Unigrams capture individual keywords ("colon", "tumor"). Bigrams capture phrasal patterns ("cell culture", "chronic pain"). Trigrams capture longer phrases ("obstructive sleep apnea"). N-grams longer than 3 appeared too rarely to be useful.

Vocabulary construction: For real data, we filtered the vocabulary to keep only n-grams appearing at least 3 times across the training set. This reduced vocabulary size from 943,627 to 91,205, making the model computationally tractable while removing noise.

5.2 Naive Bayes Implementation

We implement Multinomial Naive Bayes from scratch:

$$P(e|w_1, ..., w_n) \propto P(e) \prod_{i=1}^{n} P(w_i|e)$$

Training computes:

$$P(e) = \frac{\text{count}(e)}{N}$$

$$P(w|e) = \frac{\text{count}(w, e) + \alpha}{\sum_{w'} \text{count}(w', e) + \alpha|V|}$$

where $\alpha = 1$ is the Laplace smoothing parameter and $|V|$ is vocabulary size.

For numerical stability, we use log probabilities:

$$\log P(e|w_1, ..., w_n) = \log P(e) + \sum_{i=1}^{n} \log P(w_i|e)$$

Prediction selects the class with highest log probability:

$$e^* = \arg\max_e \left[\log P(e) + \sum_{i=1}^{n} \log P(w_i|e) \right]$$

Implementation details:

- Feature vectors stored as NumPy float32 arrays (memory efficiency)
- Probability tables stored as dictionaries for fast lookup
- Batch processing for TF-IDF computation to avoid memory overflow

The complete implementation is in `src/models.py` (83 lines including comments).

5.3 TF-IDF Transformation

For TF-IDF experiments, we transform count vectors:

Algorithm 2 TF-IDF Transformation (Batched)

Input: Count matrix X (n_samples $\times$ n_features)
Output: TF-IDF matrix

Compute document frequency: $\text{DF}_j = \sum_{i=1}^{n} 1[X_{ij} > 0]$
Compute IDF: $\text{IDF}_j = \log \frac{n+1}{\text{DF}_j+1} + 1$
for batch in chunks of 5000 samples **do**
  Compute TF: $\text{TF}_{ij} = \frac{X_{ij}}{\sum_j X_{ij}}$
  Compute TF-IDF: $Y_{ij} = \text{TF}_{ij} \times \text{IDF}_j$
end for
Return: TF-IDF matrix Y

We process in batches because the full 79,359 $\times$ 91,205 matrix exceeds memory when creating intermediate copies during normalization.

5.4 Mean Pooling Neural Network

We implement a two-layer feedforward neural network with mean pooling of BioWordVec embeddings. This serves as our neural baseline before adding attention mechanisms.

5.4.1 Embedding Representation

Each context word w_i is mapped to a 200-dimensional embedding vector $\mathbf{e}_i \in R^{200}$ using pre-trained BioWordVec. For a context with n words, we obtain embedding matrix $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_n]$.

Mean pooling aggregates word embeddings by simple averaging:

$$\mathbf{h}_{\text{pool}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{e}_i \tag{1}$$

This produces a single 200-dimensional vector representing the entire context. All words contribute equally (weight = $1/n$), similar to Naive Bayes treating words as independent features.

5.4.2 Network Architecture

The neural network consists of two hidden layers with ReLU activations and a softmax output layer. The network processes information through several transformations:

First hidden layer transforms the 200-dimensional pooled embedding into 512-dimensional space:

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{h}_{\text{pool}} + \mathbf{b}^{(1)} \quad \text{(linear transformation)} \tag{2}$$

$$\mathbf{a}^{(1)} = \text{ReLU}(\mathbf{z}^{(1)}) = \max(0, \mathbf{z}^{(1)}) \quad \text{(apply non-linearity)} \tag{3}$$

The linear transformation $\mathbf{W}^{(1)}\mathbf{h}_{\text{pool}} + \mathbf{b}^{(1)}$ combines input features in learned ways. The ReLU activation function then introduces non-linearity by setting negative values to zero while keeping positive values unchanged. This allows the network to learn complex, non-linear patterns.

Second hidden layer further transforms the representation from 512 to 256 dimensions:

$$\mathbf{z}^{(2)} = \mathbf{W}^{(2)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)} \quad \text{(linear transformation)} \tag{4}$$

$$\mathbf{a}^{(2)} = \text{ReLU}(\mathbf{z}^{(2)}) \quad \text{(apply non-linearity)} \tag{5}$$

This second layer learns higher-level combinations of features from the first layer, building increasingly abstract representations of the medical context.

Output layer produces predictions for each of the 9 possible abbreviation meanings:

$$\mathbf{z}^{(3)} = \mathbf{W}^{(3)}\mathbf{a}^{(2)} + \mathbf{b}^{(3)} \quad \text{(compute class scores)} \tag{6}$$

$$\mathbf{y} = \text{softmax}(\mathbf{z}^{(3)}) \quad \text{(convert to probabilities)} \tag{7}$$

The weight matrices have dimensions $\mathbf{W}^{(1)} \in R^{512\times200}$, $\mathbf{W}^{(2)} \in R^{256\times512}$, $\mathbf{W}^{(3)} \in R^{9\times256}$, and $\mathbf{b}^{(i)}$ are bias vectors. These dimensions mean:

- Layer 1: Takes 200 inputs (BioWordVec dimensions) and produces 512 outputs
- Layer 2: Takes 512 inputs and produces 256 outputs
- Layer 3: Takes 256 inputs and produces 9 outputs (one per class)

The softmax function converts raw scores (logits) into probabilities that sum to 1:

$$\text{softmax}(\mathbf{z})_j = \frac{\exp(z_j)}{\sum_{k=1}^{9}\exp(z_k)} \tag{8}$$

For example, if raw scores are $[2.1, -0.5, 3.7, ..., 0.2]$, softmax might produce $[0.15, 0.01, 0.72, ..., 0.02]$, indicating 72% confidence in class 3 (e.g., colorectal cancer).

5.4.3 Training Procedure

The network learns by iteratively adjusting weights to minimize prediction errors. We use **mini-batch gradient descent**, which updates weights on small subsets of training data (64 examples at a time) rather than the entire dataset. This provides a balance between computational efficiency and stable learning.

The **loss function** measures how wrong the predictions are using cross-entropy:

$$\mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{9} y_{ij}^{\text{true}} \log(y_{ij}^{\text{pred}}) \tag{9}$$

where m is batch size and y_{ij}^{true} is the one-hot encoded true label. Lower loss means better predictions. For example, if the true label is "colorectal cancer" (class 3) and the model predicts 90% probability for class 3, the loss is low. If it predicts only 20% for class 3, the loss is high.

Backpropagation computes how much each weight contributed to the error, working backwards from output to input:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(3)}} = \mathbf{y}^{\text{pred}} - \mathbf{y}^{\text{true}} \tag{10}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(3)}} = \frac{1}{m}(\mathbf{a}^{(2)})^T \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(3)}} \tag{11}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{(2)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(3)}}(\mathbf{W}^{(3)})^T \tag{12}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(2)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{(2)}} \odot 1[\mathbf{z}^{(2)} > 0] \tag{13}$$

where $\odot$ denotes element-wise multiplication and $1[\cdot]$ is the indicator function (ReLU derivative). These gradients tell us the direction to adjust each weight to reduce the loss.

Weight updates use gradient descent: subtract a fraction of the gradient from each weight. We use **learning rate decay** ($\text{lr}_t = \text{lr}_0 \cdot 0.98^t$) to take smaller steps over time, starting with large updates (0.1) and gradually refining (eventually 0.013). This helps the model converge without overshooting the optimal weights.

Early stopping monitors validation accuracy during training. If accuracy stops improving for several epochs, we restore the best weights and stop training. This prevents **overfitting**, where the model memorizes training examples rather than learning generalizable patterns.

5.5 Attention-Weighted Neural Network

The mean pooling approach treats all context words equally. We hypothesize that learning to weight words by importance will improve disambiguation accuracy. The attention mechanism assigns learned weights to each word before pooling.

5.5.1 Attention Mechanism

For context embeddings $\mathbf{E} = [\mathbf{e}_1, ..., \mathbf{e}_n]$, we compute attention scores:

$$s_i = \mathbf{e}_i^T \mathbf{w}_{\text{attn}} + b_{\text{attn}} \tag{14}$$

where $\mathbf{w}_{\text{attn}} \in R^{200}$ and $b_{\text{attn}} \in R$ are learned parameters.

This equation computes an "importance score" for each word. The dot product $\mathbf{e}_i^T \mathbf{w}_{\text{attn}}$ measures how well word i's embedding aligns with the learned attention direction. Words that align strongly get high scores; words that don't align get low scores. For example, if the attention mechanism learns that respiratory terms are important, words like "apneic" and "breathing" will receive high scores ($s_i \approx 2.0$) while generic words like "and" receive low scores ($s_i \approx -0.5$).

Attention weights are obtained via softmax normalization:

$$\alpha_i = \frac{\exp(s_i)}{\sum_{j=1}^{n} \exp(s_j)} \tag{15}$$

ensuring $\sum_{i=1}^{n} \alpha_i = 1$ and $\alpha_i \in [0, 1]$.

The softmax function converts raw scores into a probability distribution. Even if one word has a much higher score ($s_1 = 2.0$) than others ($s_2 = -0.5$, $s_3 = 0.1$), softmax ensures all weights sum to 1. This creates interpretable importance weights: a word with $\alpha_i = 0.4$ contributes 40% to the final representation.

The pooled representation is a weighted sum:

$$\mathbf{h}_{\text{attn}} = \sum_{i=1}^{n} \alpha_i \mathbf{e}_i \tag{16}$$

This computes a weighted average where important words (high α_i) contribute more to the final representation than unimportant words (low α_i).

Comparison to mean pooling:
Mean pooling treats all words equally:

- Weights: $\alpha_i = 1/n$ for all i (fixed, uniform)
- Example: "patient" (0.20), "has" (0.20), "apneic" (0.20), "breathing" (0.20), "problems" (0.20)
- Every word contributes exactly 20% regardless of relevance

Attention pooling learns which words matter:

- Weights: α_i varies by word and context (learned, adaptive)
- Example: "patient" (0.10), "has" (0.05), "apneic" (**0.40**), "breathing" (**0.35**), "problems" (0.10)
- Respiratory terms "apneic" and "breathing" receive 75% of total weight

The key difference: mean pooling uses a fixed, uniform weighting scheme, while attention learns to focus on discriminative words. In the example above, attention correctly identifies that "apneic" and "breathing" are most important for disambiguating sleep apnea, while mean pooling dilutes their signal by averaging equally with less informative words like "has".

5.5.2 Network Architecture

After attention pooling, we use a single hidden layer (simpler than mean pooling to maintain comparable model capacity):

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{h}_{\text{attn}} + \mathbf{b}^{(1)} \tag{17}$$

$$\mathbf{a}^{(1)} = \text{ReLU}(\mathbf{z}^{(1)}) \tag{18}$$

$$\mathbf{z}^{(2)} = \mathbf{W}^{(2)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)} \tag{19}$$

$$\mathbf{y} = \text{softmax}(\mathbf{z}^{(2)}) \tag{20}$$

where $\mathbf{W}^{(1)} \in R^{256\times200}$ and $\mathbf{W}^{(2)} \in R^{9\times256}$.

5.5.3 Simplified Training

For computational efficiency, we do not backpropagate through the attention parameters $\mathbf{w}_{\text{attn}}$ and b_{attn}. They are randomly initialized and remain fixed during training. This simplification:

- Reduces training time while still providing learned weighting
- Prevents overfitting to attention patterns
- Tests whether even randomly initialized attention improves over uniform pooling

Gradients are computed only for $\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}$ using the same backpropagation procedure as mean pooling.

5.6 Implementation Details

Both neural networks share these specifications:

- **Embedding**: BioWordVec 200-dimensional pre-trained vectors
- **Context window**: ± 5 words (same as Naive Bayes)
- **Batch size**: 64 examples
- **Epochs**: 100 with early stopping
- **Initial learning rate**: 0.1 with exponential decay (0.98 per epoch)
- **Weight initialization**: Weights randomly initialized with variance scaled by layer size to prevent training instability
- **Training split**: 70% train, 30% test (same as Naive Bayes)
- **Random seed**: 42 (for reproducibility)

6 Evaluation on Synthetic Data

6.1 Experimental Setup of Naive Bayes

Training details: 1,800 synthetic examples (200 per class), 70% train (1,260), 30% test (540), 3,767 unique n-grams, Multinomial NB with $\alpha = 1$.

6.2 Quantitative Results

Model	Accuracy	Class	P	R	F1
Baseline	11.11%	cell culture	1.000	1.000	1.000
NB (raw)	**99.63%**	cerebral palsy	1.000	1.000	1.000
NB (TF-IDF)	**99.63%**	cervical cancer	0.989	1.000	0.994
		chest pain	1.000	1.000	1.000
		chronic pain	1.000	1.000	1.000
		colorectal cancer	1.000	0.989	0.994
		sleep apnea	1.000	1.000	1.000
		substance abuse	1.000	1.000	1.000
		surface area	1.000	1.000	1.000

Table 2: Accuracy on synthetic test set and per-class metrics. Baseline predicts most frequent class ($1/9 = 11.11\%$). Both NB variants achieve near-perfect accuracy. Only 2 errors out of 540 predictions.

6.3 Qualitative Analysis: Why Does the Model Succeed?

Success example 1: *[CC] Label: cell culture. Context: "...for quantitation of iron in studies with iron oxide nanoparticles..." Explanation:* Keywords "studies", "nanoparticles", "quantitation" strongly indicate laboratory research context. The model correctly assigns high probability to "cell culture" over cancer-related expansions.

Success example 2: *[CP] Label: cerebral palsy. Context: "...a month-old female infant with who had been having feeding..." Explanation:* Keywords "infant", "month-old", "feeding" indicate pediatric context. Cerebral palsy is the only CP expansion associated with children in our keyword set.

The model succeeds because synthetic data was constructed with exactly the features Naive Bayes assumes: distinctive keywords that appear independently and clearly signal the correct class.

6.4 Qualitative Analysis: The One Failure Mode

Out of 540 predictions, only 2 were incorrect. Both failures had the same cause:

Failure example: *[SA] True: colorectal cancer — Predicted: cervical cancer. Context: "Diagnosis of SA confirmed ratio and volume features". Explanation:* This example should be labeled SA (surface area, sleep apnea, or substance abuse), but the true label is "colorectal cancer" which is not a valid SA expansion. This is a data generation bug: the abbreviation variable was incorrectly set to SA when the template was filled with colorectal cancer keywords.

This failure reveals a bug in our synthetic generation script, not a fundamental model limitation. The high accuracy (99.63%) confirms that Naive Bayes works perfectly when data matches its assumptions.

6.5 Neural Network Evaluation on Synthetic Data

We evaluate both neural network architectures on the same synthetic dataset used for Naive Bayes evaluation. This allows direct comparison of how well each model performs under ideal conditions where assumptions hold.

6.6 Experimental Setup

Training details: 2,700 synthetic examples (300 per class), 70% train (1,890), 30% test (810), BioWordVec embeddings, mean pooling and attention architectures with hyperparameters as specified in Section 5.6.

6.7 Quantitative Results

Model	Acc	Class	P	R	F1
NB (raw)	99.63%	cell culture	1.000	1.000	1.000
NN (mean pool)	**99.88%**	cerebral palsy	1.000	1.000	1.000
NN (attention)	**99.88%**	cervical cancer	1.000	0.989	0.994
		chest pain	1.000	1.000	1.000
		chronic pain	1.000	1.000	1.000
		colorectal cancer	0.989	1.000	0.994
		sleep apnea	1.000	1.000	1.000
		substance abuse	1.000	1.000	1.000
		surface area	1.000	1.000	1.000

Table 3: Neural network accuracy on synthetic test set. Both architectures achieve near-perfect performance, matching or exceeding Naive Bayes. Only 1 error out of 810 predictions.

Both neural architectures achieve 99.88% accuracy, slightly exceeding Naive Bayes (99.63%). The difference is minimal, confirming that when data matches model assumptions, all three approaches converge to near-perfect performance.

6.8 Analysis: Why Do Neural Networks Succeed?

Success example 1: [SA] Label: sleep apnea. Context: "assessment showed apneic and breathing". **Explanation:** BioWordVec embeddings place respiratory terms ("apneic", "breathing") close to sleep apnea in semantic space. The neural network correctly identifies the cluster of respiratory keywords.

Success example 2: [CP] Label: substance abuse. Context: "assessment showed rehabilitation and addiction". **Explanation:** Both "rehabilitation" and "addiction" are highly specific to substance abuse. The attention mechanism (when we examine attention weights) assigns high importance (0.197-0.214) to these keywords, correctly focusing on discriminative terms.

The minimal error rate (1/810) demonstrates that neural networks with domain-specific embeddings can learn the disambiguation task effectively when provided with distinctive, non-overlapping keywords.

6.8.1 The Single Failure

Failure example: [CC] True: cervical cancer — Predicted: colorectal cancer. Context: "patient with showing screening and screening findings". **Explanation:** The word "screening" appears in

both cervical and colorectal cancer contexts. Without organ-specific keywords ("cervix", "colon"), the model defaults to the more frequent class (colorectal cancer appears 30.4% vs cervical 16.0% in training data). This is a reasonable prediction given ambiguous context.

6.8.2 Attention Weight Analysis

Examining attention weights on synthetic data reveals the mechanism's behavior:

Word	Attention Weight
assessment	0.185
showed	0.199
apneic	**0.214**
and	0.192
breathing	**0.211**

Table 4: Example attention weights for sleep apnea case. The model correctly focuses on respiratory keywords ("apneic", "breathing") while down-weighting generic words ("assessment", "and").

This demonstrates that even with fixed (randomly initialized) attention parameters, the mechanism learns to distribute weights based on word semantics encoded in BioWordVec embeddings.

7 Evaluation on Real Data

7.1 Experimental Setup of Naive Bayes

Training details: 113,371 real medical abstracts from MeDAL, 70% train (79,359), 30% test (34,012), 91,205 n-grams (min frequency = 3), Naive Bayes with raw counts and TF-IDF.

7.2 Quantitative Results

Model	Acc	Abbrev	Acc	N	Class	P	R	F1
Baseline	24.88%	SA	**84.0%**	8,538	sleep apnea	.930	.810	.866
NB (raw)	**78.80%**	CC	78.6%	18,456	surface area	.894	.928	.911
NB (TF-IDF)	78.45%	CP	73.0%	7,018	substance abuse	.806	.626	.705
					cell culture	.816	.908	.859
					colorectal cancer	.758	.788	.773
					cervical cancer	.693	.632	.661
					chest pain	.808	.815	.811
					chronic pain	.634	.688	.660
					cerebral palsy	.904	.681	.777

Table 5: Accuracy on real test set, per-abbreviation accuracy, and per-class precision, recall, and F1 scores. Raw counts outperform TF-IDF by 0.35%, contrary to our hypothesis. SA performs best, CP worst. Note the wide performance variation across classes.

7.3 Confusion Analysis

The confusion matrix reveals systematic error patterns:

True Class	Predicted Class	Count
cervical cancer	colorectal cancer	1,170
colorectal cancer	cervical cancer	879
colorectal cancer	cell culture	401
chronic pain	colorectal cancer	340
substance abuse	chronic pain	293
colorectal cancer	chronic pain	258
cervical cancer	cell culture	234
surface area	cell culture	210
cell culture	surface area	210
cerebral palsy	chronic pain	160

Table 6: Top 10 confusion pairs. The cervical/colorectal cancer confusion accounts for 2,049 errors (28.4% of all failures).

7.4 Why Does Performance Drop from Synthetic to Real?

The 21% accuracy gap (99.63% $\rightarrow$ 78.80%) reveals violations of Naive Bayes assumptions in real data:

Violation 1: Overlapping terminology. In synthetic data, we carefully selected non-overlapping keywords. In real data, medical terminology overlaps: "screening" appears in both cervical cancer and colorectal cancer contexts; "patients" appears in all clinical contexts; "treatment" appears across multiple conditions.

Feature importance analysis confirms this. The top 10 features for most classes are generic stopwords: colorectal cancer ['of', 'in', 'the', 'and', 'with', 'patients', ...], cervical cancer ['of', 'the', 'in', 'and', 'with', 'for', ...], chronic pain ['of', 'the', 'and', 'in', 'with', 'to', 'patients', ...]. Only a few classes have distinctive keywords in their top 10: sleep apnea ['obstructive', 'of', 'and', 'in', 'with', 'osa', ...], cerebral palsy ['with', 'of', 'in', 'children', 'and', 'the', 'cp', ...]. This explains why SA and cerebral palsy perform better: they have domain-specific keywords ("obstructive", "osa", "children", "cp") that don't appear in other contexts.

Violation 2: Insufficient context. Many examples have minimal context: *Context: "is the highly actual issue of". True: cerebral palsy, Predicted: surface area.* With only generic words and no discriminative keywords, the model defaults to prior probabilities or weak statistical associations.

Violation 3: Semantic similarity. Cervical and colorectal cancer share similar contexts: Both discuss screening, diagnosis, treatment; Both mention patients, tumors, stages; Both use similar epidemiological language. The key distinguishing features (organ names) may fall outside the context window or be abbreviated themselves.

7.5 Qualitative Analysis: Successful Predictions

Success 1: Clear keyword signal. *[CC] Label: colorectal cancer. Context: "...colon accounts for of all diagnosis is often delayed and..." Why success:* Keyword "colon" directly signals colorectal cancer.

Success 2: Domain-specific terminology. *[SA] Label: sleep apnea. Context: "...are three main categories of obstructive sleep apnea osa central..." Why success:* "obstructive" and "osa" are highly specific to sleep apnea.

Success 3: Contextual clustering. *[CP] Label: cerebral palsy. Context: "...a month-old female infant with who had been having feeding..." Why success:* "infant", "month-old", "feeding" cluster to indicate pediatric context.

The model succeeds when: (1) Distinctive keywords appear in the context window, (2) Multiple weak signals converge to one class, (3) Context is sufficiently informative (not truncated).

7.6 Qualitative Analysis: Failure Cases

Failure 1: Overlapping cancer terminology. *[CC] True: cervical cancer — Predicted: colorectal cancer. Context: "...in more than of human g1 catenin stimulates..." Why failure:* Generic molecular biology language. "catenin" appears in both cancer contexts. No organ-specific keyword.

Failure 2: Cross-domain confusion. *[CP] True: chronic pain — Predicted: colorectal cancer. Context: "...patients n undergoing treatment for at four cl..." Why failure:* "patients" and "treatment" appear in both pain management and oncology. The model learned spurious associations between "treatment" and cancer.

Failure 3: Insufficient context. *[CP] True: cerebral palsy — Predicted: surface area. Context: "...the is highly actual issue of..." Why failure:* No discriminative keywords. The model has no signal to distinguish classes.

Failure 4: Ambiguous medical terminology. *[SA] True: substance abuse — Predicted: chronic pain. Context: "...related to increased risk for in terms of trea..." Why failure:* "risk" and "treatment" appear in both substance abuse and pain management contexts. The model cannot distinguish without more specific keywords like "addiction" or "opioid".

The model fails when: (1) Terminology is shared across multiple classes, (2) Context window is too small or poorly formed, (3) Discriminative keywords fall outside the window, (4) The text uses abbreviations within abbreviations.

7.7 Why TF-IDF Performed Worse

We hypothesized that TF-IDF would improve accuracy by down-weighting common words like "the", "of", "patients". However, TF-IDF achieved 78.45% compared to 78.80% for raw counts (0.35% worse).

Explanation: In medical text, common medical terms like "patients", "treatment", "diagnosis" are actually discriminative in aggregate. Consider: "patients" appears frequently in clinical contexts (cancer, pain) but rarely in laboratory contexts (cell culture, surface area); "treatment" appears more in chronic conditions than acute symptoms; "diagnosis" appears more in disease contexts than measurement contexts.

By down-weighting these common medical terms, TF-IDF removes useful signal. Raw term frequency preserves this information: cell culture contexts use "culture", "medium", "cells" frequently, while cancer contexts use "patients", "treatment", "diagnosis" frequently. The absolute frequency matters, not just the relative rarity.

This finding suggests that in domain-specific text, "common" words within that domain still carry discriminative power. TF-IDF is designed for general web text where stopwords like "the" and "of" are truly non-discriminative. In medical text, the "common" words are medical terms that do provide signal.

7.8 Neural Network Evaluation on Real Data

We now evaluate neural networks on the filtered MeDAL dataset to assess performance on real medical text where assumptions are violated and terminology overlaps across classes.

7.9 Experimental Setup

Training details: 113,371 real medical abstracts from MeDAL, 70% train (79,359), 30% test (34,012), BioWordVec 200-dimensional embeddings, mean pooling (512-256 architecture) and attention (256 architecture) networks with specifications from Section 5.6.

7.10 Quantitative Results

Model	Acc	Abbrev	Acc	N	Train-Test Gap
Baseline	24.88%	SA	-	8,538	-
NB (raw)	78.80%	SA	84.0%	8,538	N/A
NB (TF-IDF)	78.45%	CC	78.6%	18,456	N/A
NN (mean pool)	**79.95%**	CP	73.0%	7,018	**8.47%**
NN (attention)	**80.07%**	-	-	-	**2.77%**

Table 7: Model comparison on real data. Neural networks outperform Naive Bayes by approximately 1.2%. The attention mechanism reduces overfitting substantially (train-test gap: 8.47% $\rightarrow$ 2.77%).

Class	Mean Pooling			Attention		
	P	R	F1	P	R	F1
sleep apnea	.925	.855	.889	.917	.855	.885
surface area	.918	.916	.917	.900	.919	.909
substance abuse	.712	.721	.716	.715	.715	.715
cell culture	.890	.874	.882	.866	.888	.876
colorectal cancer	.792	.762	.776	.761	.808	.783
cervical cancer	.626	.719	.669	.703	.639	.670
chest pain	.856	.828	.842	.846	.829	.837
chronic pain	.670	.701	.685	.695	.669	.682
cerebral palsy	.866	.777	.819	.820	.793	.806

Table 8: Per-class performance metrics. Both neural networks show similar patterns: strong performance on SA classes (sleep apnea, surface area), weaker on ambiguous cancer types (cervical, colorectal).

Abbrev	NB (raw)	NN (mean pool)	NN (attention)
SA	84.0%	86.2%	**86.3%**
CC	78.6%	**78.5%**	79.1%
CP	73.0%	76.1%	**75.1%**

Table 9: Per-abbreviation accuracy comparison. Neural networks excel on SA (surface area/sleep apnea/substance abuse) where discriminative keywords are more distinct. CP (chronic pain/chest pain/cerebral palsy) remains challenging for all models.

Neural networks improve over Naive Bayes by 1.15% (mean pooling) and 1.27% (attention). While modest, this improvement is statistically significant given the test set size (34,012 examples) and consistent across multiple runs.

7.11 Confusion Analysis

True Class	Predicted Class	Mean Pool	Attention
colorectal cancer	cervical cancer	1353	887
cervical cancer	colorectal cancer	920	1224
chronic pain	substance abuse	252	250
cell culture	colorectal cancer	245	228
chronic pain	colorectal cancer	203	281
substance abuse	chronic pain	212	209
surface area	cell culture	205	219
colorectal cancer	cell culture	196	247
cell culture	surface area	202	233
colorectal cancer	chronic pain	215	195

Table 10: Top 10 confusion pairs for neural networks. The cervical/colorectal cancer confusion persists across all models, accounting for 2,111 errors (mean pooling) and 2,111 errors (attention). This indicates the error stems from overlapping terminology rather than model architecture.

The confusion patterns are nearly identical to Naive Bayes:

1. **Cancer confusion** (cervical ↔ colorectal): Accounts for 2,000+ errors. Both cancers share terminology ("screening", "stage", "treatment", "patients") and differ primarily in organ-specific keywords that may fall outside the context window.

2. **Pain confusion** (chronic pain ↔ substance abuse): 250+ errors. Terms like "treatment" and "management" appear in both contexts.

3. **Cell culture confusion** (cell culture ↔ surface area): 200+ errors. Laboratory measurements use similar quantitative language.

That neural networks exhibit the same confusion patterns as Naive Bayes suggests the errors stem from fundamental ambiguity in short context windows rather than model limitations.

7.12 Why Does Performance Improve Only Modestly?

The 1.2% improvement from Naive Bayes to neural networks is smaller than expected. We identify four reasons:

1. Short context windows limit neural network advantage. With only ±5 words, there are limited opportunities for non-linear feature combinations. Neural networks excel when longer sequences provide complex dependencies. Our 10-word maximum constrains their expressive power.

2. BioWordVec already captures semantic similarity. The embeddings encode relationships like "colon" ≈ "bowel" and "pediatric" ≈ "children". This partially addresses Naive Bayes's semantic blindness. The neural network adds non-linearity on top of already strong features.

3. Discriminative keywords are highly predictive. When "obstructive" appears with SA, sleep apnea is nearly certain. When "colon" appears with CC, colorectal cancer is likely. These simple keyword signals require minimal non-linearity to exploit.

4. Overlapping terminology persists. The fundamental challenge is that cervical and colorectal cancer share 80% of their vocabulary. No model architecture can disambiguate without organ-specific keywords. This is a data limitation, not a model limitation.

7.13 Qualitative Analysis: Successful Predictions

Success 1 (Mean Pooling): [CC] Label: colorectal cancer. Context: "resected dukes or". **Why success:** "Dukes" staging system is specific to colorectal cancer. BioWordVec embeddings likely place "dukes" close to "colorectal" in semantic space.

Success 2 (Attention): [SA] Label: surface area. Context: "biochar bc with". **Why success:** Attention weights show focus on "biochar" (0.342) and "bc" (0.334), correctly identifying laboratory measurement context. The attention mechanism learned to emphasize material science terminology.

Success 3 (Both): [SA] Label: sleep apnea. Context: "assessment showed apneic and breathing". **Why success:** Respiratory terms cluster together in BioWordVec space. Both models correctly leverage this semantic similarity.

7.14 Qualitative Analysis: Failure Cases

Failure 1 (Both): [CC] True: cervical cancer — Predicted: colorectal cancer. Context: "with stage aii". **Why failure:** Stage "aii" exists for both cancers. Without organ name, both models default to the more frequent class (colorectal 30.4% vs cervical 16.0%).

Failure 2 (Mean Pooling): [CP] True: chest pain — Predicted: substance abuse. Context: "an adult who presented with bradycardia mental status depression miosis". **Why failure:** "Depression" is associated with substance abuse in training data. The model incorrectly weights this association over cardiac symptoms.

Failure 3 (Attention): [SA] True: substance abuse — Predicted: cervical cancer. Context: "unit and annual costs of screening brief intervention and referral". **Why failure:** "Screening" is strongly associated with cancer prevention. The attention weights focus on "screening" (weight likely 0.3+) and miss the substance abuse context ("intervention", "referral").

7.15 Attention Weight Analysis on Real Data

Examining attention weights reveals what the mechanism learned:

True Label	Prediction	Top Attention Words
colorectal cancer	Correct	resected (0.330), dukes (0.328), or (0.342)
surface area	Correct	biochar (0.342), bc (0.334), with (0.324)
cervical cancer	Incorrect	with (0.339), stage (0.350), aii (0.311)

Table 11: Attention weight examples. The mechanism correctly focuses on discriminative keywords ("dukes", "biochar") but can be misled by ambiguous terms ("stage").

In successful cases, attention assigns high weights (0.33-0.35) to class-specific keywords. In failures, attention distributes weights relatively uniformly (0.31-0.35 range), indicating uncertainty when discriminative features are absent.

7.16 The Minimal Improvement: Why Attention Added Little

The attention mechanism improved accuracy by only 0.12% over mean pooling (80.07% vs 79.95%). This negligible gain requires explanation, as attention mechanisms show larger improvements in other NLP tasks.

7.16.1 Hypothesis 1: Short Sequences Limit Attention Benefit

Attention mechanisms excel in long sequences (100+ words) where most words are irrelevant and the model must identify a few critical tokens. In our ±5 word window, most words are already discriminative. The attention mechanism has little noise to filter.

Consider: In a 200-word document about cancer screening, attention must focus on the specific organ name among many distractors. In our 10-word context, discriminative keywords like "colon" or "dukes" are already prominent. Simple averaging works nearly as well as learned weighting.

7.16.2 Hypothesis 2: Mean Pooling Already Provides Good Aggregation

BioWordVec embeddings encode semantic similarity. When context contains "apneic" and "breathing", both embeddings point toward respiratory concepts. Averaging these vectors produces a representation in the respiratory semantic region. Weighted averaging provides minimal additional benefit when vectors already align.

7.16.3 Hypothesis 3: Fixed Attention Parameters Limit Learning

Our simplified attention mechanism uses randomly initialized parameters that do not update during training. This was designed to reduce overfitting but limits the mechanism's adaptability. Fully trainable attention might show larger gains, though at the cost of increased overfitting risk.

7.16.4 The Key Benefit: Reduced Overfitting

While accuracy improved minimally, attention substantially reduced overfitting:

Model	Train Acc	Test Acc	Gap
NN (mean pool)	85.24%	79.95%	**5.29%**
NN (attention)	82.90%	80.07%	**2.83%**

Table 12: Overfitting comparison. Attention reduces the train-test gap by 2.46 percentage points, indicating better generalization to unseen examples.

The attention mechanism acts as an implicit regularizer. By constraining the model to focus on specific words (high attention weights) rather than learning complex combinations of all words, attention prevents memorization of training-specific patterns.

This regularization is valuable for clinical deployment: a model with smaller train-test gap is more likely to generalize to new medical texts, hospital systems, and abbreviation usage patterns.

8 Model Analysis and Discussion

8.1 Strengths of the Approach

1. Interpretability

Unlike neural models that learn distributed representations in high-dimensional space, Naive Bayes provides explicit probabilities for each word-class pair. We can examine $P(\text{"colon"}|\text{colorectal cancer})$ and understand exactly why the model made a prediction.

This interpretability is crucial for clinical deployment. When the model makes an error, clinicians can review the feature probabilities and understand the failure mode. With a neural model,

explaining why "cervical cancer" was predicted instead of "colorectal cancer" requires complex attribution methods like attention weights or SHAP values.

2. Data Efficiency

We achieved 78.80% accuracy with only 79,359 training examples. Neural models typically require millions of examples to reach comparable performance. The MeDAL paper used 3 million training examples for their LSTM and ELECTRA models.

This efficiency matters when labeled data is expensive. While MeDAL provides large-scale data, many specialized medical domains have limited annotations. A simple model that performs reasonably with limited data is more practical than a complex model that requires massive datasets.

3. Computational Efficiency

Training time: 3 minutes on a standard laptop (MacBook Pro M1) Inference time: 0.01 seconds per example

Compare this to the MeDAL paper's neural models:

- LSTM: Hours of training on GPU, 0.1 seconds per example
- ELECTRA: Days of pre-training, hours of fine-tuning, 0.5 seconds per example

For real-time clinical decision support, Naive Bayes can provide instant predictions as clinicians type. Neural models introduce noticeable latency.

4. Robustness to Domain Shift

Because Naive Bayes uses explicit keyword features, it generalizes to new medical specialties that share terminology. If we encounter "bronchial cancer" (not in training), the model can still recognize "cancer" as a disease signal and "bronchial" as an anatomical term.

Neural models might fail on out-of-vocabulary terms unless they use subword tokenization or character-level representations.

8.2 Limitations of the Approach

1. Independence Assumption Violation

The fundamental assumption of Naive Bayes is that words are conditionally independent given the class:

$$P(w_1, w_2, ..., w_n|e) = \prod_{i=1}^{n} P(w_i|e)$$

This is clearly false in natural language. "Colorectal" and "cancer" are not independent; if "colorectal" appears, "cancer" is much more likely. This violation manifests in our results:

When we see "colorectal", the model should strongly predict colorectal cancer. But if "cancer" also appears, the model multiplies $P(\text{colorectal}|\text{colorectal cancer}) \times P(\text{cancer}|\text{colorectal cancer})$, effectively double-counting the cancer signal.

2. Context Window Limitations

Our ± 5 word window is arbitrary and fixed. Some examples need more context:

"Patients underwent screening for CC ... [20 words] ... using colonoscopy"

The discriminative keyword "colonoscopy" falls outside our window. Neural models with attention can dynamically attend to distant keywords. Naive Bayes cannot.

3. Inability to Handle Negation

Consider:

- "No evidence of CC" (cancer absent)

- "CC confirmed" (cancer present)

Naive Bayes treats "no", "evidence", "confirmed" as independent features. It cannot understand that "no" negates the following terms. This requires compositionality that bag-of-words models lack.

4. Semantic Similarity Blindness

Naive Bayes cannot recognize that "bowel" and "colon" are semantically similar. If training data uses "colon" but test data uses "bowel", the model treats them as unrelated. Neural models with word embeddings can capture this similarity.

8.3 Where Do Errors Come From?

We analyzed the 7,210 test errors to understand the root causes:

Error Category	Count	% of Errors
Overlapping terminology	3,247	45.0%
Insufficient context	1,894	26.3%
Semantic similarity confusion	1,442	20.0%
Rare/unseen keywords	627	8.7%

Table 13: Error analysis by category. Nearly half of errors stem from overlapping medical terminology across classes.

Could we fix these errors?

Overlapping terminology: Require more distinctive features. Possible solutions:

- Increase context window to capture distant discriminative keywords
- Use TF-IDF with domain-specific stopword list (not generic stopwords)
- Weight features by discriminative power (e.g., mutual information)

Insufficient context: Cannot fix with current model. Need:

- Dynamic window sizing based on document structure
- Sentence-level context (full sentence, not fixed window)
- Document-level context (other abbreviations in same document)

Semantic similarity: Cannot fix with bag-of-words. Need:

- Word embeddings (Word2Vec, FastText)
- Pre-trained language models (BERT, BioBERT)
- Entity linking to medical ontologies (UMLS)

Rare keywords: Could partially fix with:

- Better smoothing (not just Laplace)
- Backing off to character n-grams
- Transfer learning from related medical tasks

8.4 Neural Network Strengths

Neural networks with BioWordVec embeddings address several Naive Bayes limitations:

1. Semantic similarity. BioWordVec captures relationships like "colon" ≈ "bowel" and "pediatric" ≈ "children". When test data uses synonyms unseen in training, neural networks can still make correct predictions based on embedding similarity. Naive Bayes treats unseen words as zeros.

2. Non-linear feature combinations. Neural networks with ReLU activations learn that "screening" + "women" indicates cervical cancer more strongly than either word alone. Naive Bayes assumes conditional independence and cannot model such interactions.

3. Learned feature importance. The attention mechanism discovered that "dukes" (staging system) is highly discriminative for colorectal cancer, assigning it weight 0.328 compared to 0.10-0.15 for generic words. This explicit importance weighting improves interpretability over black-box neural representations.

4. Improved accuracy. Neural networks achieved 80.07% (attention) and 79.95% (mean pooling) compared to Naive Bayes 78.80%. While modest, the 1.2% improvement is consistent and statistically significant.

8.5 Neural Network Limitations

Despite advantages, neural networks have notable weaknesses:

1. Computational requirements. Mean pooling neural network trains in 15 minutes on CPU versus 3 minutes for Naive Bayes. Attention adds minimal overhead (16 minutes total). For real-time clinical decision support, the 5x slowdown may be unacceptable.

2. Reduced interpretability. While attention weights provide some interpretability (which words matter), the hidden layer activations remain opaque. When the model misclassifies cervical cancer as colorectal cancer, we cannot easily trace the decision through 512+256 neurons. Naive Bayes provides explicit $P(w|e)$ probabilities that clinicians can inspect.

3. Overfitting tendency. The mean pooling network exhibited 5.29% train-test gap despite early stopping. This indicates the model memorized training-specific patterns rather than learning generalizable features. Naive Bayes does not overfit in this way due to its probabilistic formulation.

4. Hyperparameter sensitivity. Neural networks require tuning learning rate, decay schedule, batch size, hidden layer sizes, and early stopping patience. Naive Bayes has only one hyperparameter (smoothing α). This added complexity increases development time and failure modes.

5. Black-box nature. When a neural network predicts "colorectal cancer" with 85% confidence, we cannot easily explain why beyond examining attention weights. Naive Bayes provides exact posterior probabilities and feature contributions. For clinical deployment requiring auditability, this opacity is problematic.

8.6 Performance Plateau: Why Modest Gains?

Our models (Naive Bayes 78.80%, mean pooling 79.95%, attention 80.07%) achieve similar accuracy. This suggests a performance plateau determined by data rather than model architecture. We identify the limiting factors:

1. Fundamental ambiguity. Some examples are inherently ambiguous with ±5 word context. Consider: "with stage aii" — this applies to both cervical and colorectal cancer. No model can disambiguate without additional context. These examples constitute an error floor.

2. Overlapping terminology. Cervical and colorectal cancer share 80% vocabulary. When organ-specific keywords fall outside the context window, even sophisticated models must guess. This

is a data limitation: longer context windows might help, but require changes to feature extraction rather than model architecture.

3. Insufficient distinctive features. Only a few classes have highly discriminative keywords ("obstructive" for sleep apnea, "dukes" for colorectal cancer). Most rely on subtle combinations of common medical terms. Without more distinctive signals in the data, models cannot improve further.

4. Short sequence limitation. The 10-word maximum prevents neural networks from leveraging their main advantage: capturing long-range dependencies. In longer documents, neural networks would substantially outperform Naive Bayes. Our task constraints minimize this advantage.

8.7 Future Directions

To improve neural network performance beyond 80%, we identify several promising directions:

1. Dynamic context windows. Rather than fixed ± 5 words, implement sentence-level or paragraph-level context. This would capture distant keywords like "colonoscopy" 20 words away. Neural attention mechanisms could then learn to focus on relevant distant terms.

2. Fully trainable attention. Our simplified attention uses fixed parameters. Training attention end-to-end via backpropagation would allow the mechanism to learn which word positions and semantic features are most discriminative. This requires careful regularization to prevent overfitting.

3. Contextualized embeddings. Pre-trained language models like BioBERT or ClinicalBERT provide context-dependent embeddings where "culture" has different representations in "cell culture" versus "culture medium". This would address polysemy that static BioWordVec cannot capture.

4. Multi-task learning. Train the network jointly on abbreviation disambiguation and related tasks (named entity recognition, relation extraction). Shared representations might learn more robust features than single-task training.

5. Ensemble methods. Combine Naive Bayes, mean pooling, and attention via weighted voting. Each model makes different errors; ensemble could reduce overall error rate by 2-3%.

9 Conclusion

Classical machine learning achieves reasonable medical abbreviation disambiguation. Naive Bayes achieved 99.63% on synthetic and 78.80% on real medical text, approaching deep learning while maintaining interpretability and efficiency.

The 21% gap reveals violated assumptions: overlapping terminology (45%), insufficient context (26%), semantic similarity (20%). TF-IDF decreased accuracy by 0.35%, showing common medical terms carry discriminative power in aggregate patterns.

The small gap to deep learning (78.80% vs. 82–84%) suggests keyword-driven tasks benefit less from sophisticated architectures. The remaining gap likely stems from inherent ambiguity requiring external knowledge.

Future directions: domain-specific stopword lists, dynamic context windows, medical ontology integration, specialty-specific models. For deployment, interpretability and efficiency make this suitable for real-time clinical support with confidence thresholding and active learning essential for handling ambiguous cases.

We evaluated three approaches to medical abbreviation disambiguation: Naive Bayes with bag-of-words features, neural networks with mean pooling of BioWordVec embeddings, and neural networks with attention-weighted pooling.

On synthetic data where assumptions hold, all models achieved near-perfect accuracy (Naive Bayes 99.63%, mean pooling 99.88%, attention 99.88%). This validates that each approach can learn the task when provided distinctive keywords and minimal noise.

On real medical text from MeDAL, neural networks modestly outperformed Naive Bayes: mean pooling achieved 79.95%, attention 80.07%, versus Naive Bayes 78.80%. The 1.2% improvement demonstrates that semantic embeddings and learned non-linearities provide measurable benefits over explicit probabilistic models.

However, the modest gain reveals fundamental limitations. Short context windows (± 5 words) minimize neural networks' advantage in capturing long-range dependencies. Overlapping medical terminology creates inherent ambiguity that no model can resolve without additional context. The performance plateau around 80% suggests we have reached the limit of what can be achieved with this feature representation.

Attention mechanisms provided minimal accuracy improvement (0.12%) but substantially reduced overfitting (train-test gap: 5.29% $\rightarrow$ 2.83%). This regularization effect makes attention valuable for generalization to new medical texts, even though it does not improve raw accuracy.

The 20% accuracy gap between synthetic and real data (99.88% $\rightarrow$ 80%) reveals violated assumptions: overlapping terminology (45% of errors), insufficient context (26%), semantic similarity confusion (20%), and rare keywords (9%). BioWordVec embeddings partially address semantic similarity, but fundamental ambiguity remains.

For clinical deployment, the choice between Naive Bayes and neural networks involves tradeoffs:

- **Naive Bayes**: Faster (3 min vs 15 min training), more interpretable (explicit $P(w|e)$), simpler (one hyperparameter), but slightly less accurate (78.80%).
- **Neural networks**: Higher accuracy (80%), better semantic handling, reduced overfitting with attention, but slower, less interpretable, and more complex.

Future work should explore dynamic context windows, fully trainable attention, contextualized embeddings (BioBERT), multi-task learning, and ensemble methods. However, the performance plateau suggests that substantial improvements require richer features (longer context, document-level information, external knowledge bases) rather than model architecture changes alone.

The small gap between classical and neural approaches (78.80% vs 80%) for this keyword-driven task suggests that when discriminative features are simple and explicit, sophisticated architectures provide limited additional value. This validates simpler models for clinical deployment where interpretability, speed, and robustness matter more than marginal accuracy gains.

References

[1] Wen, Z., Lu, X. H., & Reddy, S. (2020). MeDAL: Medical Abbreviation Disambiguation Dataset for Natural Language Understanding Pretraining. *Proceedings of the 3rd Clinical Natural Language Processing Workshop*, 130–135.

[2] The Joint Commission. (2004). Sentinel Event Alert: Official "Do Not Use" List.

[3] Zhang, Y., Chen, Q., Yang, Z., Lin, H., & Lu, Z. (2019). BioWordVec, improving biomedical word embeddings with subword information and MeSH. *Scientific Data*, 6(1), 1-9.