IMPERIAL

Convolution Revolution

Wiener Filters for Embedded Text Comparison

Andrei Cristian Danila
10/09/2024

Agenda

1 Introduction
2 Key Concept 1: Embeddings
3 Key Concept 2: Transformers
4 Key Concept 3: Wiener Filters
5 Wiener Loss
6 Wiener Attention
7 Conclusion

Introduction

Current text comparison methods are inadequate and reductive.
They fail to capture semantic and syntactic context due to their element-wise nature.

Using Wiener Filters, I propose a novel text comparison method.

Experiments:
- Wiener Loss in Transformer (Translation)
- Wiener Attention in Transformer (Sentiment Classification)

The results show potential for breaking state-of-the-art benchmarks.

Key Concept 1

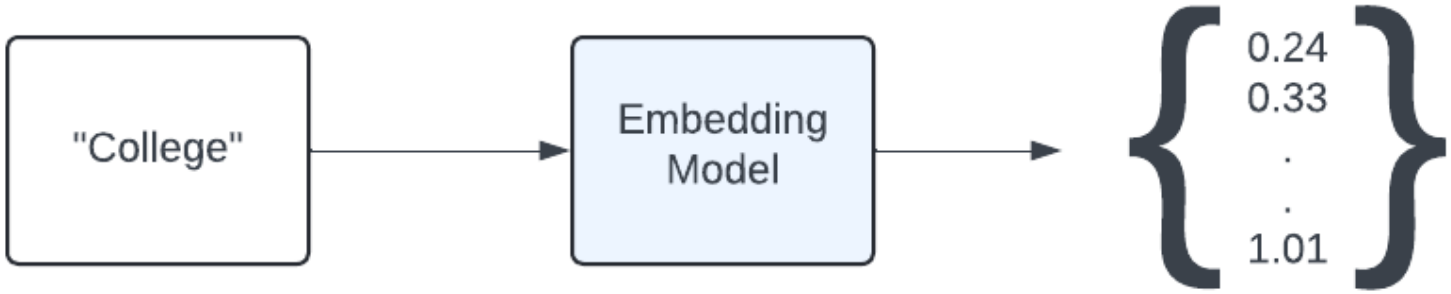
Embeddings

Embeddings
Text to vector

Embeddings are vector representations of words/subwords.
They capture semantic and syntactic meaning in high-dimensional space.

- Intuitively: similar words will be close in the embedding space.

How do we create embeddings?
With predefined models such as Word2Vec [1], or using learned layers (BERT or GPT).



Key Concept 2

Transformers

Transformers
Attention is All You Need

The Transformer is an NLP architecture that revolutionized Machine Learning.
Introduced in “Attention is All You Need” [2], it solved most of the shortcomings of recurrent models (RNN), while increasing parallelization and compute efficiency.

- You know it from: ChatGPT, Claude, Gemini.

The Attention Mechanism
The main novelty was the Attention mechanism, which allowed calculating similarity between tokens regardless of distance, solving the vanishing gradient problem. At the end of the attention layer, the output is a set of enriched (contextualized) embeddings.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

Key Concept 3

Wiener Filters

Wiener Filters

From Signal Processing to Text Comparison

Originally introduced for denoising and deblurring [3], the Wiener Filter is repurposed.
The Wiener filter minimizes mean squared error (MSE) to restore degraded signals, particularly effective in the frequency domain. The filter is derived by solving the optimization problem:

$$\min_{v} \frac{1}{2} \|Yv - x\|^2$$

where **v** is the Wiener filter, **x** is the signal, and **Y** represents convolution with another signal **y**. For efficient computation, the solution is expressed in the Fourier domain as:

$$v(x, y) = \mathcal{F}^{-1}\left(\frac{\mathcal{F}\{x\} \otimes \mathcal{F}\{y\} + \lambda}{\mathcal{F}\{x\} \otimes \mathcal{F}\{x\} + \lambda}\right)$$

with *F* as the fast Fourier transform (FFT), ⊗ as the Hadamard product, and λ as a regularization parameter.

Wiener Filters

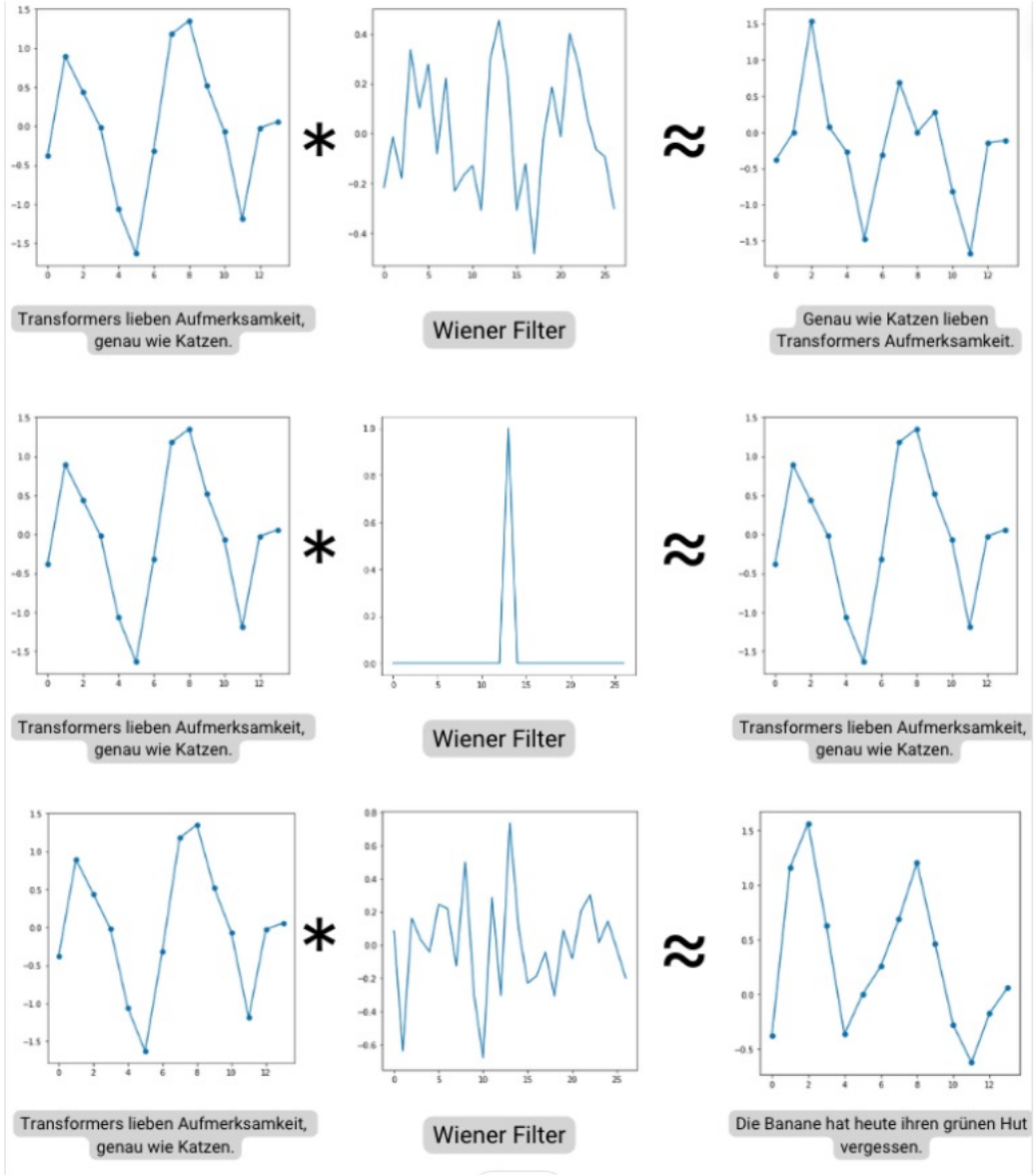
How do we adapt them for text?

We need to think of text as a signal.
Embedded text can be sliced either on the sequence axis or on the embedding axis.

- **Sequence axis (Dimension-by-Dimension)**
  - We interpret **each embedding dimension** of a sequence/sentence as a signal.
- **Embedding axis (Token-by-Token)**
  - We compare two tokens across **all their embeddings.**

A		
Dogs	**love**	**attention**
0.36	0.09	0.65
0.24	0.17	0.90
0.41	0.09	0.07
0.19	0.23	0.46
0.39	0.47	0.75
⋮	⋮	⋮
0.37	0.75	0.40

B		
Cats	**love**	**attention**
0.30	0.83	0.51
0.01	0.06	0.32
0.54	0.24	0.76
0.57	0.96	0.53
0.57	0.36	0.83
⋮	⋮	⋮
0.12	0.05	0.54



Wiener Loss

Implementation and results

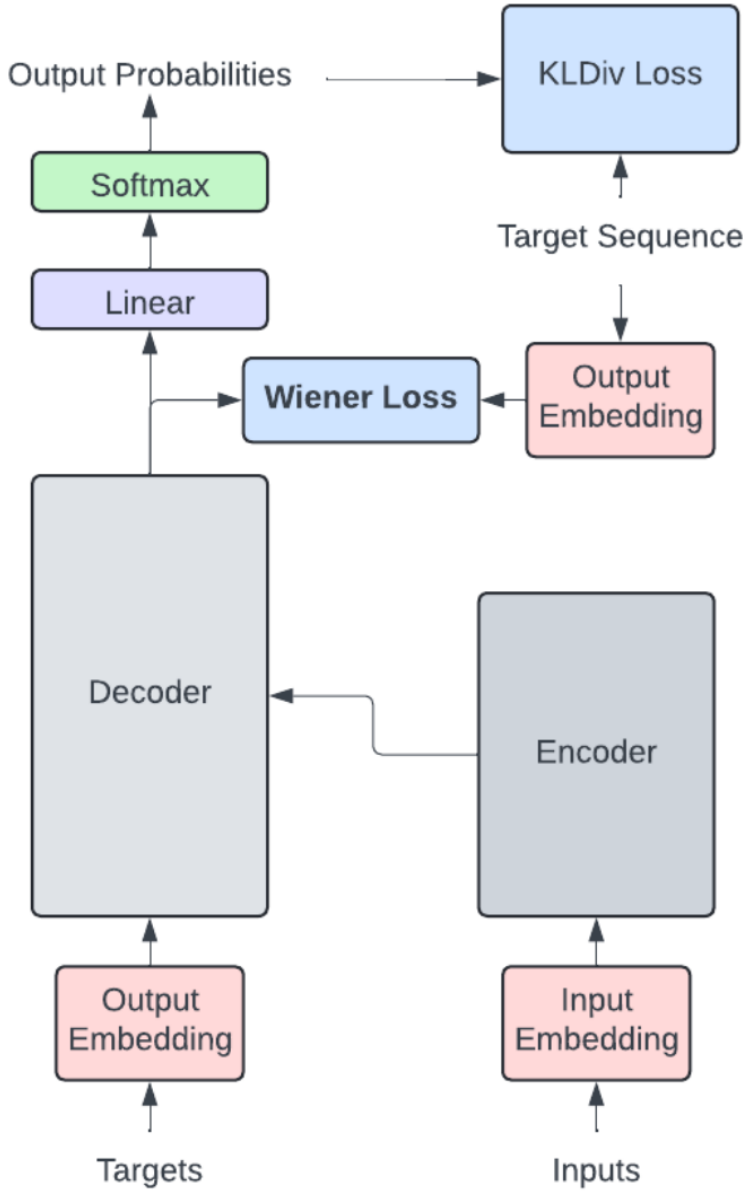
Wiener Loss
Dimension-by-Dimension

"Convolve and Conquer" [4] formulated the Wiener Loss as:

$$\mathcal{L}_W(\mathbf{x}_\theta, \mathbf{y}) = \frac{1}{2} \|\mathbf{W}\{\mathbf{v}(\mathbf{x}_\theta, \mathbf{y}) - \delta\}\|^2$$

In this case, x_θ and y are corresponding embedding dimensions of the target sequence (e.g. comparing dimension 1 of the target and the output). v is the filter that matches x_θ and y, and δ is the identity of the convolution operation.

- The Wiener Loss is applied pre-classification head, and acts as a **regularization** loss. The model is still able to enrich the embeddings, but overfitting is prevented through Wiener regularization.



Wiener Loss

Training and Results

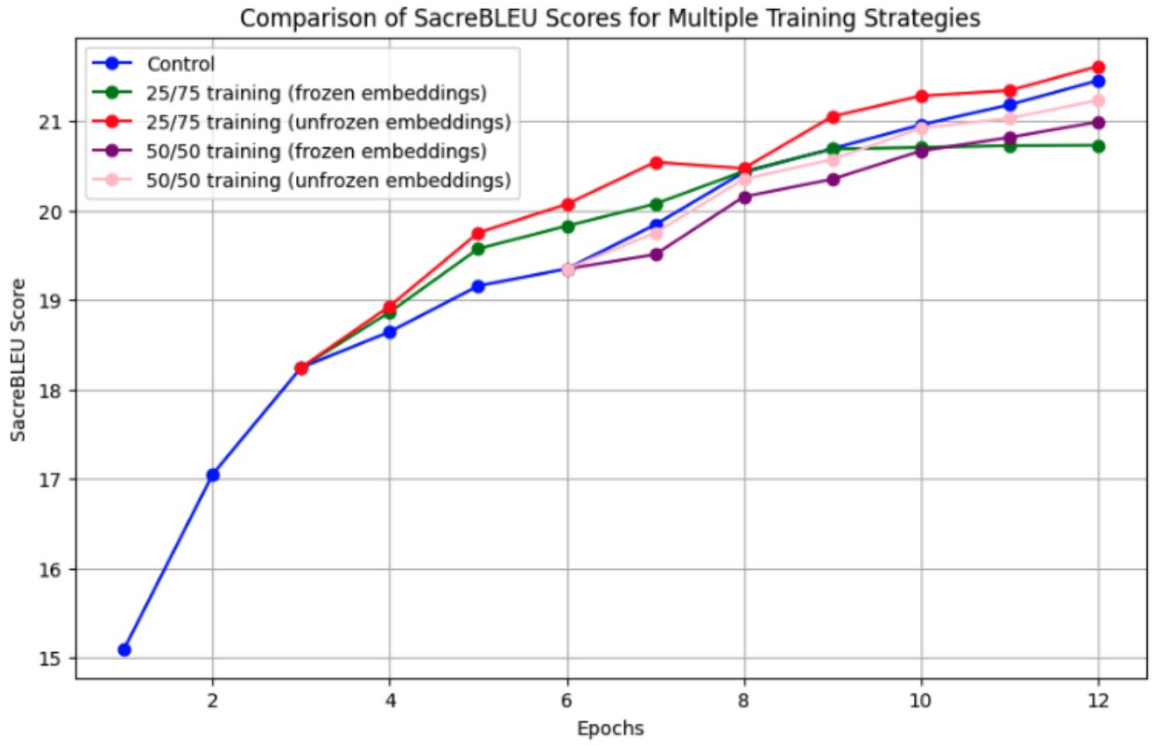
Initial training didn't succeed, as the Wiener Loss over-regularised the model
To combat this, I implemented **two-stage** training.

- The model would train without the Wiener Loss for a certain number of epochs, after which I introduced the Wiener Loss.

I was able to surpass the control model, for both a short and a long training schedule.

Freezing the embeddings: I experimented with freezing the embedding weights in the second stage.
Training: 1 epoch - 1 hour on 8xA100. No discernible overhead from Wiener Loss.

Variant	Embedding Type	Epochs	SacreBLEU
Control	Learned	6	19.35
Single Stage	Learned	6	18.91
Two Stage (50/50 split)	Learned *(frozen in second stage)*	6	19.94
Two Stage (50/50 split)	Learned *(not frozen)*	6	**20.08**
Control	Learned	12	21.45
Two Stage (50/50 split)	Learned *(frozen in second stage)*	12	20.99
Two Stage (25/75 split)	Learned *(frozen in second stage)*	12	20.73
Two Stage (50/50 split)	Learned *(not frozen)*	12	21.23
Two Stage (25/75 split)	Learned *(not frozen)*	12	**21.62**



Wiener Attention

Implementation and results

Wiener Attention
Token-by-Token

Using Wiener Similarity, the Attention mechanism is redefined as:

$$\text{Wiener Attention}(Q, K, V) = \text{softmin}\left(\frac{WSM(Q, K)}{\sqrt{d_k}}\right) V$$

where Q, K and V remain the keys, but the similarity between Q and K is now calculated using the Wiener Similarity. *softmin* is used, as Wiener Similarity is 0 when the match is perfect.

- Wiener Attention calculates similarity between Q and K convolutionally, taking into context all embedding dimensions.

Classic Attention Matrix

	$\vec{Q}_1$	$\vec{Q}_2$	$\vec{Q}_3$	$\vec{Q}_4$
$\vec{K}_1$	$K_1 \cdot Q_1$	$K_1 \cdot Q_2$	$K_1 \cdot Q_3$	$K_1 \cdot Q_4$
$\vec{K}_2$	$K_2 \cdot Q_1$	$K_2 \cdot Q_2$	$K_2 \cdot Q_3$	$K_2 \cdot Q_4$
$\vec{K}_3$	$K_3 \cdot Q_1$	$K_3 \cdot Q_2$	$K_3 \cdot Q_3$	$K_3 \cdot Q_4$
$\vec{K}_4$	$K_4 \cdot Q_1$	$K_4 \cdot Q_2$	$K_4 \cdot Q_3$	$K_4 \cdot Q_4$

Wiener Attention Matrix

	$\vec{Q}_1$	$\vec{Q}_2$	$\vec{Q}_3$	$\vec{Q}_4$
$\vec{K}_1$	$WSM_{1,1}$	$WSM_{1,2}$	$WSM_{1,3}$	$WSM_{1,4}$
$\vec{K}_2$	$WSM_{2,1}$	$WSM_{2,2}$	$WSM_{2,3}$	$WSM_{2,4}$
$\vec{K}_3$	$WSM_{3,1}$	$WSM_{2,2}$	$WSM_{3,3}$	$WSM_{3,4}$
$\vec{K}_4$	$WSM_{4,1}$	$WSM_{4,2}$	$WSM_{4,3}$	$WSM_{4,4}$

Wiener Attention
Training and Results

Computational overhead

Wiener Similarity is one order of magnitude more time-costly then Dot Product. As the time complexity of the Transformer scales quadratically, this resulted in extreme overhead.

- As a proof-of-concept, I used a BERT [5] model, with a single layer and single headed attention, for a sentiment classification task (IMDb movie reviews).

The results are mostly comparable with the classic Attention mechanism, however further testing needs to be conducted.

Model Name	Eps	Gamma	Epoch	Precision	Accuracy
Classic	N/A	N/A	2	0.707	**0.707**
Wiener 1	1e-05	0.1	1	**0.710**	0.695
Wiener 2	1e-05	0.2	2	0.692	0.683
Wiener 3	1e-05	0.3	2	0.698	0.697
Wiener 4	1e-04	0.1	3	0.689	0.689
Wiener 5	1e-04	0.2	2	0.698	0.698
Wiener 6	1e-04	0.3	2	0.699	0.695
Wiener 7	1e-03	0.1	1	0.700	0.696
Wiener 8	1e-03	0.2	1	0.695	0.695
Wiener 9	1e-03	0.3	2	0.696	0.685

Table 4: Model Performance with Best Accuracy and Precision

IMPERIAL

Thank you

Convolution Revolution:
Wiener Filters for Embedded Text Comparison
10/09/2024

References

- [1]: Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *In Proceedings of the International Conference on Learning Representations (ICLR),* 2013.

- [2]: Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

- [3]: Norbert Wiener. Extrapolation, Interpolation, and Smoothing of Stationary Time Series: With Engineering Applications. *The MIT Press*, 1949.

- [4]: Deborah Pelacani Cruz, George Strong, Oscar Bates, Carlos Cueto, Jiashun Yao, and Lluis Guasch. Convolve and conquer: Data comparison with Wiener Filters, 2023.

- [5]: Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Bidirectional encoder representations from transformers. *arXiv preprint arXiv:1810.04805, 2018*