SOFTMAX FUNCTION IN A SELF-ATTENTION MECHANISM

Allaberdiev B.
National University of Uzbekistan, Tashkent, Uzbekistan
allaberdiyev_91@mail.ru

A machine translation technique called neural machine translation (NMT) uses artificial neural networks to predict the probability of word sequences. Typically, it incorporates all sentence models into one integrated model. This is the dominant approach today[1, 2]and under certain conditions can produce translations that compete with human translations when translating between languages with few resources[3]. However, especially in languages where high-quality data is less available[1, 4, 5] and there are still problems with domain switching between the data the system is trained on and the text to be translated. available[1]. NMT systems also tend to produce literal translations[5].

The Transformer model quickly became the dominant choice for machine translation systems [2] and was the most used architecture so far at the 2022 and 2023 Workshop on Statistical Machine Translation [6, 7].

The transformer model consists of two parts, and the main work of both parts is performed by the multi-head attention mechanism.

The self-attention mechanism is a fundamental component of the Transformer model, which is widely used in natural language processing tasks. Here's a detailed explanation of how it works, including an example with the calculation of softmax.

Step-by-Step Explanation

1. Input Representation:

- Each word in the input sentence is represented as a vector. For simplicity, let's assume these vectors have been embedded into a fixed-size vector space.

2. Creating Query, Key, and Value Vectors

- For each word in the input sentence, we create three vectors: Query (Q), Key (K), and Value (V). These vectors are obtained by multiplying the embedding vector by trained matrices W_Q, W_K and W_V. The dimensionality of these vectors is typically chosen to be smaller than the embedding vector for computational efficiency[8].

- If the input vector dimension is d_{modeld}, the dimension of Q, K, and V vectors is typically smaller (e.g., 64). This is an architectural choice to reduce computational complexity.

3. Calculating Scores

- To compute the self-attention for a particular word, say "Thinking" at position 1, we calculate a score for each word in the input sentence by taking the dot product of the query vector of the word in position 1 (q_1) with the key vectors of all words ($k_1, k_2, \ldots, k_n$)[9].

This can be represented as:

$$score(i,j) = Q_i * K_j$$

- For example, if we are calculating the self-attention for the word at position 1 (denoted as Q1), we need to compute the dot product of Q1 with K1, Q1 with K2, and so on.

4. Scaling the Scores:

- The scores are divided by the square root of the dimension of the Key vectors (denoted as $\sqrt{d_k}$). This helps to prevent the gradients from becoming too small during backpropagation.

$$scaled_{score(i,j)} = \frac{score(i,j)}{\sqrt{d_k}}$$

5. Applying Softmax:

- The scaled scores are passed through a softmax function to obtain the attention weights. The softmax function converts the scores into probabilities that sum to 1.

$$attention\_weight(i,j) = softmax\left(\frac{score(i,j)}{\sqrt{d_k}}\right)$$

6. Calculating the Weighted Sum:

- Each Value vector is multiplied by its corresponding attention weight, and the results are summed to obtain the final output vector for each word.

$$output_i = \sum_j (attention\_wight(i,j) * V_j)$$

Example:

- Score for word at position 1: score$_1$=$q_1 \cdot k_1$=112
- Score for word at position 2: score$_2$=$q_1 \cdot k_2$=96

Dividing by $\sqrt{d_k}$:

The scores are divided by the square root of the dimensionality of the key vectors (d_k). This step helps to stabilize the gradients during training by scaling the dot products. In this example, d_k=64, so $\sqrt{d_k}$=8.

- Adjusted score for word at position 1: $\frac{score_1}{\sqrt{d_k}} = \frac{112}{8} = 14$
- Adjusted score for word at position 2: $\frac{score_2}{\sqrt{d_k}} = \frac{96}{8} = 12$

Applying this to our adjusted scores:

- Let $x_1 = 14$ and $x_2 = 12$
- First, compute the exponentials:

$e^{14} \approx 1.2026 \times 10^6$, $e^{12} \approx 1.6275 \times 10^5$

- Sum of exponentials:

$e^{14} + e^{12} \approx 1.2026 \times 10^6 + 1.6275 \times 10^5 \approx 1.3654 \times 10^6$

Softmax probabilities:

$$softmax_1 = \frac{e^{12}}{e^{14}+e^{12}} \approx \frac{1.226 \times 10^6}{1.3654 \times 10^6} \approx 0.88$$

$$softmax_2 = \frac{e^{12}}{e^{14}+e^{12}} \approx \frac{1.6275 \times 10^5}{1.3654 \times 10^6} \approx 0.12$$

Thus, after applying softmax, the probabilities are:

Score for word at position 1: 0.88

Score for word at position 2: 0.12

Summary

1. **Creating Vectors**: Generate Query, Key, and Value vectors for each word.
2. **Calculating Scores**: Compute dot products between the query vector of the current word and key vectors of all words.
3. **Dividing by $\sqrt{d_k}$**: Scale the scores by the square root of the dimensionality of the key vectors.
4. **Applying Softmax**: Convert the scaled scores into probabilities using the softmax function, which normalizes them.

This process helps in determining how much focus to put on each word in the sentence when encoding the current word.

References

1. Koehn, Philipp (2020). Neural Machine Translation. Cambridge University Press.
2. Stahlberg, Felix (2020-09-29). “Neural Machine Translation: A Review and Survey”.
3. M.Popel, M.Tomkova, J.Tomek, L.Kaiser, J.Uszkoreit, O.Bojar, Z.Zabokrtsky,(2020-09-01). "Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals". Nature Communications. 11 (1): 4381. ISSN 2041-1723.
4. B.Haddow, R.Bawden, B.Miceli, V.Antonio, J.Helcl, A.Birch(2022). "Survey of Low-Resource Machine Translation". Computational Linguistics. 48 (3): 673–732.
5. T.Poibeau, N.Calzolari, F.Bechet, P.Blache, K.Choukri, Ch.Cieri, T.Declerck, S.Goggi, H.Isahara, B.Maegaard. "On "Human Parity" and "Super Human Performance" in Machine Translation Evaluation". Proceedings of the Thirteenth Language Resources and Evaluation Conference. Marseille, France: European Language Resources Association: 6018–6023.
6. T.Kocmi, R.Bawden, O.Bojar, A.Dvorkovich, Ch.Federmann, M.Fishel, T.Gowda, Y.Graham, R.Grundkiewicz, B.Haddow, R.Knowles, P.Koehn, Ch.Monz, M.Morishita, M.Nagata, L.Barrault, O.Bojar, F.Bougares, R.Chatterjee, M.Costa-jussa, Ch.Federmann, M.Fishel, A.Fraser. Findings of the 2022 Conference on Machine Translation (WMT22). Proceedings of the Seventh Conference on Machine Translation (WMT). Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics. pp. 1–45.
7. T.Kocmi, E.Avramidis, R.Bawden, O.Bojar, A.Dvorkovich, Ch.Federmann, M.Fishel, M.Freitag, T.Gowda, R.Grundkiewicz, B.Haddow, P.Koehn, B.Marie, Ch.Monz, M.Morishita. Findings of the 2023 Conference on Machine Translation (WMT23): LLMs Are Here but Not Quite There Yet. Proceedings of the Eighth Conference on Machine Translation. Singapore: Association for Computational Linguistics. pp. 1–42.

8. Pavan Belagatti. Understanding the Softmax Activation Function: A Comprehensive Guide/ Intelligent applications, now at a lakehouse near you. Watch the product launch on demand. March, 2024
9. Hunter Philips. A Simple Introduction to Softmax. / Medium.com. May 2023.