IJSCIA

International Journal of Scientific Advances
ISSN: 2708-7972
Volume: 4 | Issue: 4 | Jul – Aug 2023 Available Online: www.ijscia.com
DOI: 10.51542/ijscia.v4i4.1

Comparative Analysis of Hidden Markov Model and Bidirectional Long Short-Term Memory for POS Tagging in Eastern Armenian

Varuzhan H. Baghdasaryan

Armenia, Yerevan, Romanos Melikian 6/1, National Polytechnic University of Armenia

*Corresponding author details: Varuzhan H. Baghdasaryan; varuzh2014@gmail.com

ABSTRACT

This scientific article introduces a novel POS tagging system specifically developed for the Eastern Armenian language. The primary objective of the study is to conduct a comparative analysis of two well-established methods for part-of-speech (POS) tagging: the hidden Markov model (HMM) coupled with the Viterbi algorithm, and an artificial neural network in the form of a recurrent neural network. The study places particular emphasis on the Eastern Armenian language and employs the ArmSpeech-POS Eastern Armenian part-of-speech tagged corpus for conducting comprehensive experiments and evaluations. POS tagging is a fundamental task in natural language processing (NLP) that involves assigning grammatical tags to words in a given text. Accurate POS tagging is crucial for various NLP applications, including machine translation, information retrieval, and sentiment analysis. The Viterbi algorithm is a well-established probabilistic method that utilizes a hidden Markov model (HMM) to determine the most likely sequence of POS tags. On the other hand, RNNs, a type of deep learning model, can capture complex patterns and dependencies in sequential data. Experimental results indicate that both methods achieve reasonable accuracy in POS tagging for Eastern Armenian. However, the RNN outperforms the Viterbi algorithm, exhibiting higher accuracy rates. This can be attributed to the RNN's ability to capture long-range dependencies and learn intricate linguistic patterns. The article concludes by discussing the implications of the study's findings and potential areas for further research. It emphasizes the significance of accurate POS tagging for improving NLP applications in Eastern Armenian and suggests exploring advanced neural network architectures and incorporating linguistic features to enhance POS tagging performance.

Keywords: part-of-speech tagging; Eastern Armenian; Viterbi algorithm; recurrent neural network; ArmSpeech-POS dataset; natural language processing; hidden Markov model.

INTRODUCTION

Part-of-speech (POS) tagging is a technique used in natural language processing (NLP) to assign grammatical tags or labels to words in a sentence. These tags indicate the role of each word in the sentence, such as whether it is a noun, verb, adjective, adverb, preposition, or conjunction. POS tagging is used in a variety of applications in NLP, including text classification, information retrieval, and machine translation. For example, in named entity recognition, POS tags are used to identify proper nouns, aiding in the identification of named entities. In sentiment analysis, POS tags help identify adjectives and adverbs, assisting in determining sentiment. In automatic summarization, POS tags aid in identifying crucial sentence components. In language translation, POS tags help identify sentence grammar, enabling accurate translation. In summary, POS tagging is a fundamental tool in natural language processing, enhancing the accuracy and efficiency of various language processing tasks.

There are several methods used for POS tagging. The main of them are below [1]:

- Rule-Based methods.
- Stochastic methods.
- Transformation-Based Learning (TBL) methods.
- Artificial Neural Network-Based methods.
- Hybrid tagging methods.

Rule-based methods use handcrafted linguistic rules to assign POS tags to words based on their context [1, 2]. These rules are typically created by linguistic experts and rely on patterns, regular expressions, and lexical information to make tagging decisions. Rule-based approaches are interpretable and can handle domain-specific rules effectively. For example, if a word ends in "ing", it is likely a verb in the present participle form.

Stochastic methods employ statistical models to estimate the probability of a word being assigned a particular POS tag based on its context [1]. Hidden Markov models, Conditional Random Fields (CRFs), and Maximum Entropy Markov Models (MEMMs) are commonly used in stochastic POS tagging. These models consider the sequence of words and the corresponding tags to compute the most likely tag sequence for a given sentence [1].

Transformation-Based Learning is a machine learning approach that learns transformational rules to convert an initial tagging to a correct one iteratively [1]. It starts with an initial set of hand-tagged data and applies a set of rules to update the tags based on contextual information. This process continues until a stopping criterion is met.

Neural network-based part-of-speech tagging is a computational approach that utilizes artificial neural networks to automatically assign grammatical labels to words in a given text and leverages the power of deep learning algorithms to capture complex patterns and dependencies within the sequential structure of language [1, 3]. The neural network architecture commonly used for POS tagging includes layers such as the embedding layer, recurrent neural network (RNN) layer (e.g., Long Short-Term Memory, Gated Recurrent Unit), and a fully connected layer.

The hybrid tagging method involves combining two or more of the above methods to improve the accuracy and robustness of the POS tagging system [1, 2]. For example, a rule-based system can be combined with a statistical model to improve accuracy.

The choice of method for POS tagging depends on the specific application and the availability of annotated data. A rule-based method is typically faster and simpler to implement, but may not perform as well as statistical and hybrid methods on complex or ambiguous text. Statistical and hybrid methods, on the other hand, require large amounts of annotated data to train the machine-learning models but can achieve higher accuracy and robustness.

Related Works:

Several studies have been conducted in the field of part-of-speech tagging, aiming to improve the accuracy and efficiency of POS tagging systems. This section provides an overview of some notable related works and techniques in the area of POS tagging.

In the context of the Armenian language, the research on POS tagging datasets and systems is relatively limited. However, the paper "ArmSpeech-POS: Eastern Armenian Part-of-Speech Tagged Corpus" provides insights into some relevant studies in this area [4].

Armtreebank GitHub repository showcases a parsing tool designed specifically for Eastern Armenian text, providing a range of functionalities including lemmatization, part-of-speech tagging, morphological feature analysis, and dependency parsing using ArmTreeBank's tokenizer module [5]. As stated by the author(s), the author(s) employ a neural network known as COMBO to carry out the tasks of lemmatization, part-of-speech tagging, and dependency parsing.

The COMBO network has undergone training using the ArmTDP treebank, which comprises around 500 sentences. The part-of-speech tagging achieved an accuracy rate of 85.07%.

Timofey Arkhangelskiy has developed a Python library that serves as an Eastern Armenian morphological analyzer [6]. This analyzer, available as a GitHub repository, offers a range of features and tools for analyzing the morphological aspects of Eastern Armenian text. In this morphological analyzer for modern Eastern Armenian, a rule-based approach is employed. It utilizes a formalized representation of literary Eastern Armenian morphology, encompassing dialectal elements, and leverages the “uniparser-morph” tool for parsing. The analyzer conducts a comprehensive morphological analysis of Eastern Armenian words, encompassing lemmatization, part-of-speech tagging, grammatical tagging, and glossing.

Another study conducted by Chahan Vidal-Gorène and Bastien Kindt focuses on the application of a joint learning approach for lemmatization and POS-tagging in Classical Armenian, Old Georgian, and Syriac languages [7]. The dataset used for Classical Armenian consists of 66,812 tokens, with 16,417 unique tokens sourced from three different corpora. The dataset underwent a division into three distinct subsets: the training subset encompassing 80% of the data, the validation subset representing 10% of the data, and the test subset also comprising 10% of the data. For the POS-tagging task, both Conditional Random Field (CRF) and linear decoder techniques were employed [7]. In the training phase, the CRF achieved an accuracy of 94.03%, while the linear decoder achieved an accuracy of 94.85%.

Dataset

As previously mentioned, the research employed the ArmSpeech-POS tagged corpus to train the POS taggers. According to the "ArmSpeech-POS: Eastern Armenian Part-of-Speech Tagged Corpus" paper, the dataset developed in the frames of this study consists of 6,081 sentences, totaling 57,160 tagged tokens [4]. It is worth noting that the dataset follows the naming conventions of the Penn Treebank and Universal Dependencies tagsets, with two versions available. The total number of tags is 16 (VERB, VERB_PLURAL, AUXILIARY_VERB_PLURAL, PUNCTUATION_MARK, and so on).

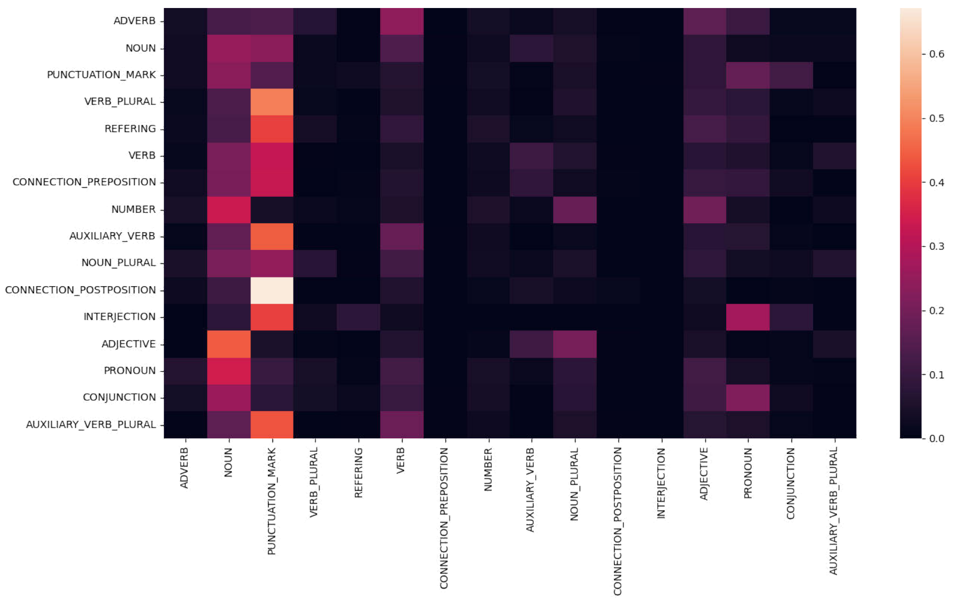

FIGURE 1: The heatmap of dataset tags.

The dataset was partitioned into training and testing sets for both HMM and RNN training. In the case of HMM, the split followed an 80/20% ratio. The training set encompassed 4,864 sentences, with a total of 45,780 tokens and 12,081 unique words. Conversely, the testing set comprised 1,217 sentences, encompassing 11,380 tokens.

For RNN training, a similar split was employed, with 80% of the dataset designated for training, including a 20% subset for validation, and the remaining 20% utilized as the testing set.

METHODS

As mentioned earlier, this study aims to develop and evaluate two distinct approaches for POS tagging. The first method entails utilizing the hidden Markov model (combined with the Viterbi algorithm), while the second method involves employing an RNN-based neural network. These two techniques will be thoroughly compared and analyzed in the context of POS tagging.

The hidden Markov model is a statistical model widely used for part-of-speech tagging in natural language processing. It is based on the underlying assumption that the POS tags of a sequence of words can be modeled as a Markov process, where each word's POS tag depends only on its own tag and the previous tag in the sequence.

The HMM consists of two main components: the hidden states and the observed emissions. In the context of POS tagging, the hidden states represent the POS tags, while the observed emissions represent the words in the sentence [8]. The goal of the HMM is to estimate the most likely sequence of hidden states (POS tags) given the observed sequence of emissions (words).

The HMM assumes two fundamental probabilities: the transition probability and the emission probability [8]. The transition probability refers to the probability of transitioning from one POS tag to another, and it is typically estimated from a large corpus of tagged data. The emission probability refers to the probability of observing a specific word given a particular POS tag. The elements comprising a hidden Markov model are shown in Figure 2.

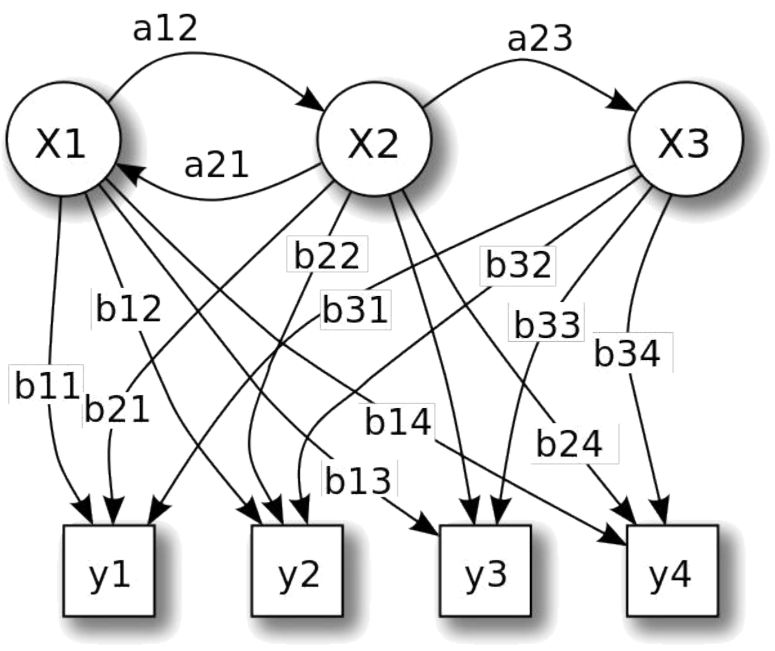

FIGURE 2: The elements comprising a hidden Markov model.

In Figure 2 **X** is the set of all **n** possible states (**X = {x_1, x_2, ..., x_n}**), **Y** is the set of all possible **m** observations (**Y = {y_1, y_2, ..., y_m}**), **A** is the state transition probabilities matrix (**A = {a_{ij}}**), where each **a_{ij}** represents the probability of moving from the state **i** to state **j**), and **B** is the probabilities of emitting an observation given a particular state (**B = {$b_i(y)$}**, where **$b_i(o)$** represents the probability of emitting observation **o** from state **i**) [8, 9].

The decoding algorithm used for HMMs is called the Viterbi algorithm. By utilizing the hidden Markov model in conjunction with the Viterbi algorithm, POS tagging can be performed effectively [9]. The HMM captures the probabilistic relationships between POS tags and observed words, while the Viterbi algorithm efficiently finds the most likely sequence of POS tags given the observed words.

This combination is widely used in natural language processing tasks and provides accurate POS tagging results [9].

The Viterbi algorithm is a statistical tagging algorithm used to find the most likely sequence of hidden states (POS tags) given the observed sequence of emissions (words) [10, 11].

It is a dynamic programming algorithm that efficiently computes the maximum probability path through the HMM. The algorithm keeps track of the most likely path for each state at each time step, considering both the transition probabilities and the emission probabilities [10, 11]. By backtracking through the computed paths, the algorithm can determine the most likely sequence of POS tags for the given input sentence (see Figure 3).

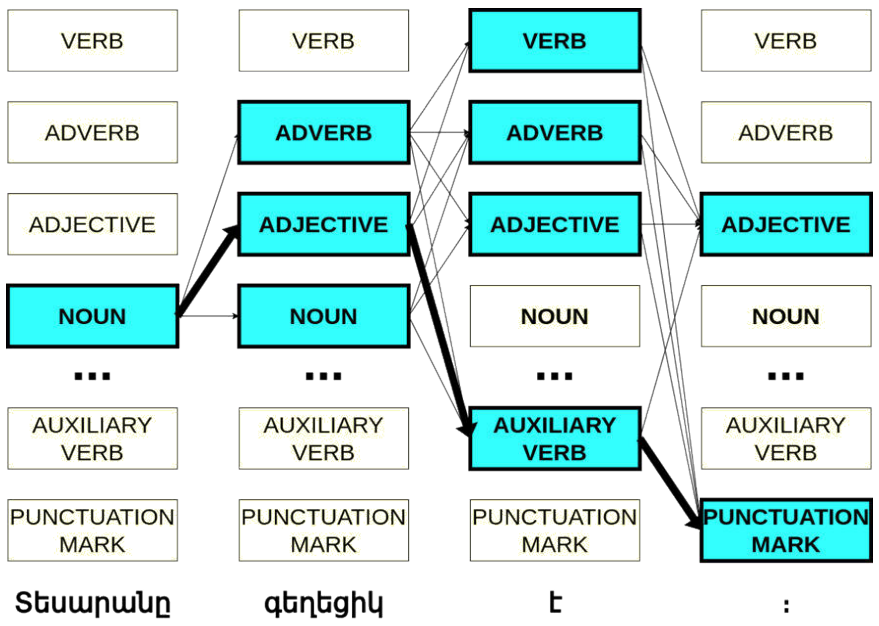

FIGURE 3: The illustration of the decoding process using the Viterbi algorithm.

Listed below are the fundamental steps and accompanying descriptions outlining the integration of the hidden Markov model with the Viterbi algorithm to accomplish accurate part-of-speech tagging [8, 9, 10, 11]:

- Hidden Markov model for POS tagging:
  - ➢ Define the set of states as POS tags and the set of observations as words or tokens.
  - ➢ Determine the initial probabilities, which represent the probabilities of starting with each POS tag.
  - ➢ Define the transition probabilities, which represent the probabilities of transitioning from one POS tag to another.
  - ➢ Determine the emission probabilities, which represent the probabilities of observing a word given a particular POS tag.
  - ➢ Use a labeled training corpus to estimate the model parameters (initial, transition, and emission probabilities) through techniques like maximum likelihood estimation or expectation maximization.

- Viterbi algorithm for POS tagging:
  - ➢ Given an input sentence or sequence of words, initialize the Viterbi trellis, which is a dynamic programming table.
  - ➢ Set the initial probabilities in the trellis using the initial probabilities from the HMM.
  - ➢ For each subsequent word in the sequence, calculate the Viterbi probabilities for each POS tag at the current position by considering the transition probabilities and the emission probabilities.
  - ➢ Update the Viterbi trellis by selecting the highest probability path to each POS tag at the current position.
  - ➢ Repeat the preceding two steps for each subsequent word in the sequence.
  - ➢ Terminate by selecting the highest probability path in the last position.
  - ➢ Trace back the optimal path by following the pointers that were used to update the Viterbi trellis, starting from the highest probability path in the last position and moving backward to the first position.
  - ➢ Output the sequence of POS tags corresponding to the optimal path found by the Viterbi algorithm.

The analysis compares the performance of the Viterbi algorithm, a well-established probabilistic method that utilizes a hidden Markov model, with that of an artificial neural network.

There are several widely used artificial neural network-based methods for part-of-speech tagging, including Multilayer Perceptron (MLP), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and others.

RNNs, as a type of deep learning model, have shown remarkable capabilities in capturing complex patterns and dependencies in sequential data [13]. RNNs are well-suited for modeling sequential data, making them effective in language-related tasks such as POS tagging [13, 14]. In this approach, the input to the RNN model is a sequence of words or tokens, and the goal is to predict the corresponding POS tags for each word. The RNN processes the input sequence one word at a time while maintaining an internal hidden state that captures the contextual information from previous words. This hidden state is updated recursively as the RNN processes each word, allowing it to learn and encode the sequential information. The RNN-based POS tagging model typically consists of an embedding layer, an RNN layer, and a fully connected layer [12]. The embedding layer maps each word to a continuous vector representation, allowing the model to capture the semantic and syntactic information of the words [12]. The RNN layer, often implemented as a Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU), processes the embedded words and maintains the hidden state and finally, the fully connected layer predicts the POS tags for each word based on the learned representations [12].

The effectiveness of the RNN-based POS tagging method depends on the quality and size of the training data. It requires a large annotated dataset with word-tag pairs to learn the patterns and correlations between words and their corresponding POS tags. Additionally, feature engineering techniques such as word embeddings, character-level representations, or linguistic features can be incorporated to enhance the performance of the model. The RNN-based POS tagging method has been widely applied and has demonstrated state-of-the-art performance in various languages and domains. It benefits from the ability of RNNs to capture long-range dependencies and contextual information, making it effective in accurately assigning POS tags to words in a given sentence.

Training and Results

In the introduction section, the article highlights the significance of utilizing hybrid models in POS tagging, which combine the strengths of different approaches. In line with this, the study employs the Viterbi algorithm, a well-established and widely used method for POS tagging. The Viterbi algorithm, in its vanilla form, is employed to assign POS tags to words in a sentence, without explicitly considering the challenge of unknown words. Unknown words, referring to words present in the test set but not encountered during the training phase, pose a challenge for accurate tagging. To address this limitation, a modified version of the algorithm is introduced, specifically designed to handle unknown words by leveraging other tagging models. To mitigate this, various techniques can be employed. The complexity of the Armenian language presents difficulties in implementing rule-based models, making them less suitable for accurate POS tagging. Therefore, in this article, a probabilistic-based model is employed in combination with the Viterbi algorithm to address the challenges posed by unknown words in Armenian POS tagging. The model leverages the occurrence probabilities of POS tags, which are derived from the training data. By assigning weights based on the probability of tag occurrence, the transition probabilities of tags are adjusted, allowing for more informed predictions for unknown words.

The Python programming language was used to implement the vanilla Viterbi algorithm, which achieves an accuracy of 75% after training on the Eastern Armenian dataset.

The modified version of the algorithm, which addresses the tagging of unknown words, demonstrates improved performance with an accuracy of 81.25%.

By employing the Viterbi algorithm in combination with a probabilistic-based model and considering the occurrence probabilities of POS tags, this study aims to enhance the accuracy of POS tagging in the Eastern Armenian language, particularly when dealing with unknown words. The results of the experiment provide valuable insights into the effectiveness of the modified Viterbi algorithm for addressing this challenge in POS tagging tasks.

For neural network-based training, the focus was specifically on the bidirectional LSTM (BiLSTM) network, which is an extension of the traditional LSTM (Long Short-Term Memory) network and is considered a highly effective network architecture and often regarded as the state-of-the-art approach for solving POS tagging tasks. LSTM is a type of RNN architecture that addresses the vanishing gradient problem by introducing memory cells and gating mechanisms. BiLSTM networks enhance the capability of LSTM networks by incorporating bidirectional processing, where the input sequence is processed in both forward and backward directions [14]. This enables the model to capture not only the past context but also the future context of each token in the sequence (see Figure 4).

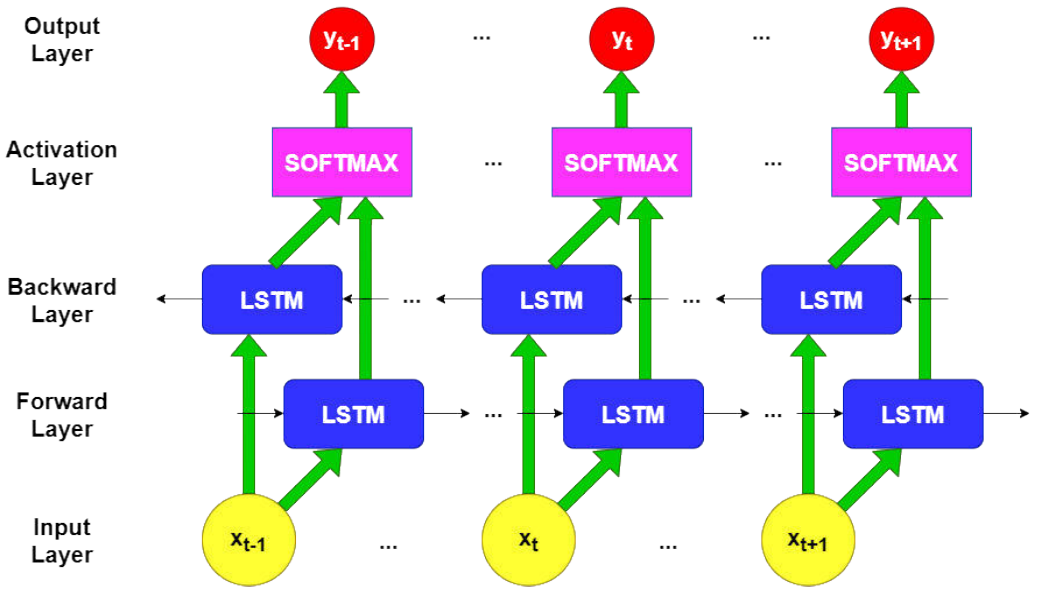

FIGURE 4: The structure of bidirectional LSTM.

To simplify the training and testing process, the implementation of the bidirectional LSTM provided by the Keras deep learning library was utilized. To determine the optimal number of LSTM hidden layers, an experimental analysis was conducted, testing different sizes including 256, 512, 1024, and 2048. Based on a thorough comparison of the experimental results, it was determined that the LSTM hidden layer with a size of 1024 yielded the best performance.

For a detailed illustration of the neural network structure, please refer to Figure 5. This figure provides a comprehensive visualization of the architecture used in the experiment, showcasing the arrangement of layers and their connections for POS tagging.

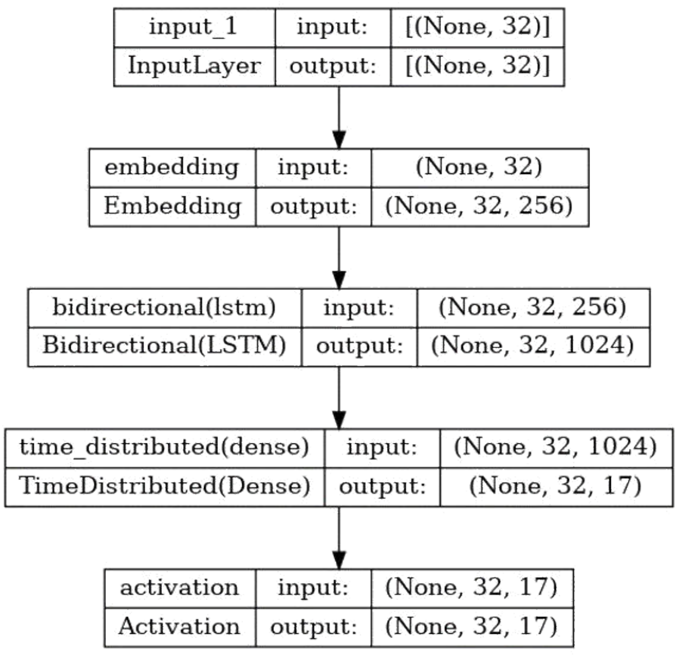
input_1 | input: | [(None, 32)]
InputLayer | output: | [(None, 32)]
embedding | input: | (None, 32)
Embedding | output: | (None, 32, 256)
bidirectional(lstm) | input: | (None, 32, 256)
Bidirectional(LSTM) | output: | (None, 32, 1024)
time_distributed(dense) | input: | (None, 32, 1024)
TimeDistributed(Dense) | output: | (None, 32, 17)
activation | input: | (None, 32, 17)
Activation | output: | (None, 32, 17)

FIGURE 5: Final model structure with 6,010,129 trainable parameters.

The embedding layer calculates word vector representations for the words present in the dataset. It constructs a word embedding model that captures the semantic and syntactic properties of the words, enabling the network to learn meaningful word representations. The dense layer (or Fully-Connected Layer) is responsible for selecting the appropriate POS tag for each word. By applying a mapping function, this layer leverages the learned features to assign POS tags. Since the dense layer operates on each element of the sequence individually, the TimeDistributed modifier is employed to ensure that the layer processes the entire sequence. In addition to these components, the neural network employs the Softmax function as the final activation function. The Softmax function normalizes the output of the network into a probability distribution over all possible POS tags, ensuring that the predicted tag probabilities sum up to 1. This allows for the selection of the most probable POS tag for each word in the sequence. By utilizing the Softmax function, the neural network provides a probabilistic interpretation of the POS tagging task, enabling more robust and accurate predictions. The neural network in this study was trained using the Adam optimizer, a popular and effective optimization algorithm widely used in deep learning.

After 5 epochs of training on the Eastern Armenian dataset, the BiLSTM neural network reached an accuracy of 95.3%. Figure 6 illustrates the progression of loss and accuracy metrics throughout the epochs. It provides a visual representation of how these performance indicators evolve over time, offering insights into the model's learning process and its ability to make accurate predictions.

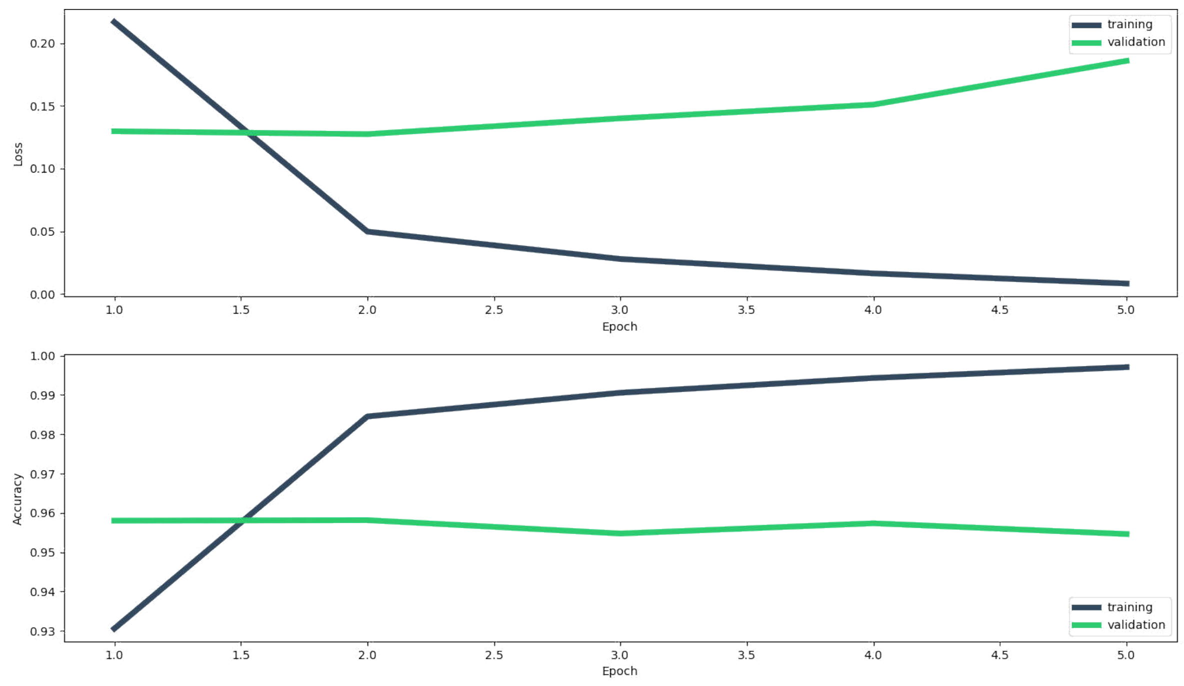

FIGURE 6: The progression of loss and accuracy metrics throughout the epochs.

CONCLUSIONS

This study encompassed an experiment centered around part-of-speech tagging, utilizing the ArmSpeech-POS Eastern Armenian part-of-speech tagged corpus. The primary aim revolved around evaluating the efficacy of two distinct POS tagging methods: the hidden Markov model (combined with the Viterbi algorithm) and a bidirectional LSTM-based recurrent neural network (RNN).

The dataset comprised 6,081 annotated sentences, with a total of 16 POS tags. The Viterbi algorithm was modified to handle unknown words by assigning weights based on the probability of tag occurrence and adjusting the transition probabilities of tags. On the other hand, the RNN model was trained using a bidirectional LSTM neural network.

The experimental findings demonstrated that the conventional Viterbi algorithm attained an accuracy rate of 75.0%, whereas the modified Viterbi algorithm designed to handle unknown words achieved an accuracy of 81.25% in POS tagging. However, the bidirectional LSTM-based RNN exhibited remarkable performance, surpassing both Viterbi approaches with a significantly higher accuracy of 95.33%. These results highlight the superior effectiveness of the bidirectional LSTM in capturing the complex linguistic patterns and context dependencies present in the Eastern Armenian language.

As a result of this study, a new Eastern Armenian POS tagger has been developed, based on the bidirectional LSTM neural network architecture. This tagger demonstrates impressive accuracy in assigning POS tags to Eastern Armenian text, showcasing its potential for various natural language processing applications.

The implications of this study's findings are significant for Eastern Armenian language processing and related fields. The high accuracy achieved by the developed POS tagger opens up opportunities for more accurate text analysis, language understanding, and machine translation in Eastern Armenian. This new tool can contribute to advancements in Eastern Armenian language technology and support various language-related tasks.

While this study focused on evaluating the Viterbi algorithm and RNN-based POS tagging methods, there are several avenues for further research. First, exploring advanced neural network architectures, such as transformer-based models, could enhance the accuracy and robustness of POS tagging. Additionally, investigating the impact of data augmentation techniques and incorporating contextual embeddings, such as BERT, could lead to improved performance. Furthermore, extending the study to other languages and domains would provide insights into the generalizability of the proposed methods. In conclusion, this study demonstrated the effectiveness of the bidirectional LSTM-based RNN model for POS tagging in Eastern Armenian, outperforming the traditional Viterbi algorithm. The development of a new Eastern Armenian POS tagger further adds value to the field. These findings contribute to the advancement of POS tagging techniques and have practical implications for natural language processing applications in the Eastern Armenian language. Further research can explore advanced architectures and incorporate additional linguistic features to enhance the performance of POS tagging systems in diverse language contexts.

REFERENCES

[1] Deepika Kumawat, Vinesh Jain. (May 2015). POS Tagging Approaches: A Comparison. International Journal of Computer Applications (0975 – 8887), Volume 118 – No. 6.

[2] Asif Ekbal, Samiran Mandal, Sivaji Bandyopadhyay. (2007). POS Tagging using HMM and Rule-based Chunking. The Proceedings of SPSAL, 8(1), 25-28.

[3] Helmut Schmid. (1994). Part-of-Speech Tagging With Neural Networks. In COLING 1994 Volume 1: The 15th International Conference on Computational Linguistics, Kyoto, Japan.

[4] Varuzhan Baghdasaryan. (2023). ArmSpeech-POS: Eastern Armenian Part-of-Speech Tagged Corpus. International Journal of Scientific Advances (IJSCIA), Volume 4| Issue 2: Mar-Apr 2023, Pages 265-270, URL: https://www.ijscia.com/wp-content/uploads/2023/04/Volume4-Issue2-Mar-Apr-No.426-265-270.pdf.

[5] Armtreebank. (24.12.2018). End-to-end Parser. Retrieved from https://github.com/Armtreebank/End-to-end-Parser.

[6] timarkh. (21.11.2021). Eastern Armenian morphological analyzer. Retrieved from https://github.com/timarkh/uniparser-grammar-eastern-armenian.

[7] Chahan Vidal-Gorène and Bastien Kindt. (2020). Lemmatization and POS-tagging process by using joint learning approach. Experimental results on Classical Armenian, Old Georgian, and Syriac. In Proceedings of LT4HALA 2020 - 1st Workshop on Language Technologies for Historical and Ancient Languages, pages 22–27, Marseille, France. European Language Resources Association (ELRA).

[8] Sean R Eddy. (2004). What is a hidden Markov model?. Nature biotechnology, 22(10), 1315-1316.

[9] Denis Eka Cahyani, Mtchael Juan Vindiyanto. (2019, November). Indonesian part of speech tagging using hidden Markov model–Ngram & Viterbi. In 2019 4th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE) (pp. 353-358). IEEE.

[10] G.D. Forney. (1973). The viterbi algorithm. Proceedings of the IEEE, 61(3), 268-278.

[11] H.-L. Lou. (1995). Implementing the Viterbi algorithm. IEEE Signal processing magazine, 12(5), 42-52.

[12] Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, 132306.

[13] Yin, W., Kann, K., Yu, M., & Schütze, H. (2017). Comparative study of CNN and RNN for natural language processing. arXiv preprint arXiv:1702.01923.

[14] Peilu Wang, Yao Qian, Frank K. Soong, Lei He, Hai Zhao. (2015). Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. arXiv preprint arXiv:1510.06168.