Applications of the Viterbi Algorithm and Comparisons with Alternative Sequence Inference Methods

Tudor Gulin

February 1, 2026

Abstract

This project explores the Viterbi algorithm as a general method for decoding the most probable hidden state sequence in Hidden Markov Models (HMMs). A generic implementation of the Viterbi algorithm is presented and applied to three distinct domains: weather prediction, text typo correction, and part-of-speech (POS) tagging. In each case, the hidden states, observations, and probabilistic model parameters are explicitly defined. The Viterbi algorithm is further compared with alternative approaches, including greedy decoding, forward–backward inference, CRF-style max-sum inference, and classical string similarity methods.

1 Introduction

Many real-world problems involve reasoning over sequential data where the underlying system states are not directly observable. Hidden Markov Models (HMMs) provide a principled probabilistic framework for such problems by modeling a sequence of hidden states that emit observable outputs. Given a sequence of observations, one of the central tasks is *decoding*: finding the most likely sequence of hidden states that explains the observations.

The Viterbi algorithm is a dynamic programming method that efficiently solves this decoding problem. In this project, a generic implementation of the Viterbi algorithm is developed and applied to multiple domains in order to demonstrate its generality and limitations.

2 Hidden Markov Models

An HMM is defined by the tuple (E, F, π, P, Q), where:

- E is a finite set of hidden states,
- F is a finite set of observations,
- $\pi(s) = P(x_0 = s)$ is the initial state distribution,
- $P(s', s) = P(x_t = s \mid x_{t-1} = s')$ is the transition probability matrix,
- $Q(s, o) = P(y_t = o \mid x_t = s)$ is the emission probability matrix.

Given an observation sequence $y_{0:T-1}$, the decoding problem consists of finding:

$$x^*_{0:T-1} = \arg\max_{x_{0:T-1}} P(x_{0:T-1} \mid y_{0:T-1}).$$

3 The Viterbi Algorithm

The Viterbi algorithm solves the decoding problem using dynamic programming. It introduces the quantity:

$$\delta_t(s) = \max_{x_{0:t-1}} P(x_{0:t-1}, x_t = s, y_{0:t}),$$

which represents the probability of the most likely partial path ending in state s at time t.

3.1 Initialization

For $t = 0$, the algorithm initializes:

$$\delta_0(s) = \pi(s) \cdot Q(s, y_0).$$

3.2 Recursion

For $t = 1, \ldots, T-1$, the recurrence relation is:

$$\delta_t(s) = \max_{s' \in E} \left[\delta_{t-1}(s') \cdot P(s', s) \cdot Q(s, y_t) \right].$$

At each step, a backpointer is stored to record which previous state maximized the expression.

3.3 Termination and Backtracking

The final state is chosen as:

$$x^*_{T-1} = \arg\max_{s \in E} \delta_{T-1}(s).$$

The optimal path $x^*_{0:T-1}$ is then recovered by following the stored backpointers backward in time.

3.4 Complexity

The time complexity of the Viterbi algorithm is $\mathcal{O}(|E|^2 \cdot T)$, which is significantly more efficient than enumerating all possible state sequences.

4 Generic Implementation

In this project, the Viterbi algorithm is implemented in a generic form that operates solely on:

- a sequence of observations encoded as indices,
- an initial probability vector π,
- a transition matrix P,
- an emission matrix Q.

This design allows the same implementation to be reused across different application domains, with only the model parameters and interpretation of states and observations changing.

5 Application 1: Weather Prediction

5.1 Model Definition

In the weather prediction task, the hidden states represent weather conditions:

$$E = \{\text{Hot}, \text{Cold}\} \quad \text{(or extended to four states)}.$$

The observations correspond to the number of ice creams consumed:

$$F = \{1 \text{ ice cream}, 2 \text{ ice creams}, 3 \text{ ice creams}\}.$$

The transition matrix encodes the likelihood of weather changes between days, while the emission matrix models the relationship between weather and ice cream consumption.

5.2 Objective

Given a sequence of observed ice cream consumptions, the goal is to infer the most likely sequence of weather states. The true hidden states are generated synthetically and later ignored, allowing the inferred Viterbi path to be compared against ground truth.

6 Application 2: Typo Correction

6.1 Model Definition

In the typo correction task, the model operates at the character level. The hidden state x_t represents the intended character, while the observation y_t represents the mistyped character.

$$E = F = \{\texttt{a}, \ldots, \texttt{z}, \texttt{space}\}.$$

6.2 Transition Model

The transition probabilities are derived from a character-level language model trained on text data. This captures contextual information such as which characters are likely to follow one another.

6.3 Emission Model

The emission matrix models keyboard noise based on a QWERTY layout. Correct characters receive high probability, neighboring keys receive moderate probability, and all other characters receive small non-zero probability to avoid impossible paths.

6.4 Interpretation of Viterbi Output

The Viterbi algorithm reconstructs the most likely sequence of intended characters. It does not explicitly predict words; instead, words emerge implicitly as sequences of characters. This formulation corresponds to a classical noisy-channel model for spelling correction.

7 Application 3: Part-of-Speech Tagging

7.1 Model Definition

For POS tagging, the hidden states correspond to grammatical tags:

$$E = \{\text{NOUN}, \text{VERB}, \text{ADJ}, \ldots\}.$$

The observations are words from a fixed vocabulary extracted from training data.

7.2 Training Phase

The HMM parameters are estimated from labeled sentences:

- π is estimated from sentence-initial tags,
- P from tag bigram frequencies,
- Q from word–tag co-occurrence frequencies.

7.3 Decoding

Given a sentence of words, the Viterbi algorithm is used to infer the most likely sequence of POS tags. The results are compared against those produced by a neural tagger (spaCy) for reference.

8 Comparison with Other Methods

Several alternative approaches are implemented for comparison:

8.1 Greedy Decoding

A naive greedy method selects, at each time step, the state that maximizes the emission probability. This approach ignores temporal dependencies and serves as a baseline.

8.2 Forward–Backward (MPM)

The forward–backward algorithm computes marginal posterior probabilities for each time step. Selecting the most probable state independently at each position yields the Marginally Maximized Path (MPM), which may differ from the globally optimal Viterbi path.

8.3 CRF-Style Inference

A max-sum inference procedure in log-space is implemented to mimic conditional random field decoding. While the graphical structure is identical to an HMM, the interpretation differs due to the use of log-linear scores.

8.4 String Similarity Methods

For typo correction, classical dictionary-based approaches are implemented:

- Weighted Levenshtein distance,
- Jaro–Winkler similarity.

These methods operate locally on individual words and do not model sequence-level context.

9 Conclusion

This project demonstrates that the Viterbi algorithm is a versatile and reusable method for sequence decoding across diverse domains. While modern neural models often outperform HMM-based approaches, Viterbi remains a valuable baseline due to its interpretability, determinism, and clear probabilistic foundations.