Hệ Thống Dự Đoán Năng Lượng Tiêu Thụ Real-Time Sử Dụng Công Nghệ Dữ Liệu Lớn

Phạm Đức Thể[1,2,3,4], Trần Thành Luân[1,2,3,4], Mai Đức Thuận[1,2,3,4], and Đỗ trọng Hợp[1,2,3,5]

[1] Đại học Quốc gia TP. HCM, Việt Nam
[2] Đại học Công nghệ Thông tin TP. HCM, Việt Nam
[3] Khoa Khoa học và Kỹ thuật Thông tin
[4] {19522253,19521810,19522316}@gm.uit.edu.vn
[5] hopdt@uit.edu.vn

Tóm tắt nội dung Ngày nay, việc sử dụng năng lượng là một vấn đề quan trọng đối với kinh tế và môi trường. Việc sử dụng năng lượng không hiệu quả có thể dẫn đến tăng giá năng lượng và ô nhiễm môi trường. Do đó, cần phải sử dụng năng lượng một cách hiệu quả và tiết kiệm nhằm bảo vệ môi trường và giảm chi phí sử dụng năng lượng. Trong báo cáo này, chúng tôi thực hiện xây dựng một hế thống dự báo mức tiêu thụ năng lượng real-time trên bộ dữ liệu *Smart Home Dataset with Weather Information*. Chúng tôi đã tiến hành xử lí dữ liệu, sau đó áp dụng các phương pháp dự báo chuỗi thời gian đơn biến và đa biến với các mô hình: TCNForecaster, Seq2SeqForecaster, LSTMForecaster, CNN, GRU, LSTM. Chúng tôi sử dụng độ đo RMSE để đánh giá hiệu suất của các mô hình dự đoán. Kết quả tốt nhất mà chúng tôi đạt được với RMSE là 0.489 sử dụng LSTMForecaster với phương pháp chuỗi thời gian đa biến sử dụng thuộc tính nhiệt độ và độ ẩm (temperature+humidity). Sau đó, chúng tôi tiến hành xây dựng hệ thống dự đoán năng lượng tiêu thụ theo thời gian thực.

Keywords: Energy Consumption Prediction · Gated Recurrent Units · Convolution Neural Network · Long Short-Term Memory · BigDL · Chronos Forecaste · Apache Spark · Apache Kafka · Kafka Streaming.

1 Giới Thiệu

Năng lượng đóng vai trò quan trọng trong việc tăng trưởng kinh tế và bảo vệ môi trường. Do sự phát triển của công nghiệp cùng với sự phát triển của kinh tế xã hôi làm cho nhu cầu sử dụng năng lượng tăng lên nhanh chóng. Việc này có thể gây ra những tác động xấu đến môi trường, bao gồm tăng nhiệt độ và ô nhiễm không khí. Nhiều nước đang chuyển đổi sang nguồn năng lượng tái tạo và tự nhiên, xây dựng các hệ thông dự báo, thay đổi quy trình sản xuất của mình và giảm thiểu tác động đến môi trường.

Hệ thống dự đoán năng lượng tiêu thụ là một hệ thống sử dụng các thiết bị IoT để thu thập dữ liệu về năng lượng tiêu thụ của các thiệt bị hằng ngày trong

đời sống và sử dụng các giải thuật dự đoán để xác định mức năng lượng tiêu thụ dự kiến trong tương lai. Kết quả của hệ thống này có thể được sử dụng để tối ưu hóa việc sử dụng năng lượng, giảm chi phí, giảm tác động tiêu cực đến sức khỏe còn người và môi trường.

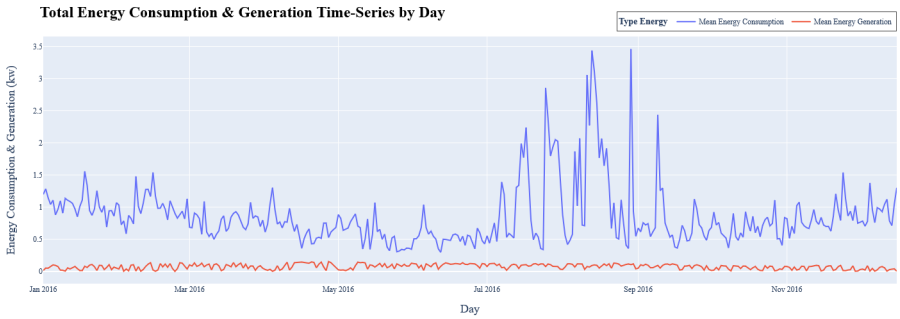
Trong báo cáo này, trước tiên chúng tôi trình bày bộ dữ liệu *"Smart Home Dataset with Weather Information"* trong Phần 2. Hướng tiếp cận của hệ thống sẽ được mô tả chi tiết ở Phần 3. Ở Phần 4, chúng tôi sẽ tiến hành thực nghiệm với các mô hình và phương pháp khác nhau. Sau đó, so sánh kết quả đạt được để chọn ra mô hình tốt nhất để xây dựng hệ thống. Tiếp theo, chúng tôi sẽ tiến hành xây dựng hệ thống ở Phần 5. Cuối cùng, chúng tôi đưa ra kết luận và hướng phát triển ở Phần 6.

2 Bộ Dữ Liệu

Bộ dữ liệu được sử dụng trong hệ thống dự đoán năng lượng này là *Smart Home Dataset with Weather Information (SHDWI)* và được thu thập từ Kaggle - được biết đến một nền tảng trực tuyến cho cộng đồng Khoa học dữ liệu, cho phép người dùng chia sẻ, tìm kiếm các bộ dữ liệu. Bộ dữ liệu có 32 cột và hơn 500,000 điểm dữ liệu với khoảng thời gian cập nhật 1 phút/lần về năng lượng được sử dụng (tính bằng kW) bởi các thiết bị của ngôi nhà thông minh và điều kiện thời tiết của khu vực vào một thời điểm nhất định. Bộ dữ liệu ghi lại thông tin tiêu thụ năng lượng của căn nhà thông minh từ 05:00 AM ngày 01/01/2016 cho đến 03:29 AM ngày 16/12/2016. Dựa vào nội dung các cột của bộ dữ liệu, chúng tôi chia dữ liệu thành 2 nhóm:

- Dữ liệu năng lượng: Mức tiêu thụ năng lượng của từng phòng/thiết bị mỗi phút, được tính bằng kW
  - Living Room, Kitchen, Home Office: Những cột này ghi mức sử dụng năng lượng của từng phòng trong nhà tính bằng kW.
  - Fridge, Barn, Dish Washer, Microwave, Furnace, Wine Cellar, Garage Door, Well: Số đọc của đồng hồ đo năng lượng của các thiết bị riêng lẻ được sử dụng trong nhà vào một ngày cụ thể.
  - Generator, House overall: Ttổng mức sử dụng năng lượng của tất cả các thiết bị và tổng năng lượng do máy phát tạo ra vào một ngày cụ thể.
- Dữ liệu thời tiết: Các thuộc tính có liên quan đến điều kiện thời tiết bao gồm, nhiệt độ, độ ẩm, lượng mưa, điểm sương, khả năng hiển thị, v.v.
  - Temperature, Humidity, Apparent Temperature: Đây là nhiệt độ và độ ẩm được ghi lại trong ngày tại địa điểm đó.
  - Visibility, Dew Point, Wind Bearing, Wind Speed, Pressure: Phần này chứa tốc độ gió và các yếu tố khí hậu khác trên địa điểm đó ở các tỷ lệ khác nhau.

Hình 1 thể hiện tổng năng lượng tạo ra và tiêu thụ theo từng ngày của bộ dữ liệu. Chúng ta có thể thấy mức năng lượng tiêu thụ trong ngày cao hơn đáng kể so với mức năng lượng được tạo ra. Ngoài ra, năng lượng tiêu thụ cũng biến động lớn hơn so với năng lượng được tạo ra theo các ngày/tháng trong năm.



Hình 1: Năng lượng tạo ra và tiêu thụ theo từng ngày của bộ dữ liệu.

Chúng tôi tiến hành chia bộ dữ liệu SHDWI thành 2 tập là Training và Streaming. Tập dữ liệu Training chiếm 80% của bộ dữ liệu SHDWI dùng để training và testing các mô hình. Tập dữ liệu Streaming chiếm 20% của bộ dữ liệu SHDWI dùng để giả lập streaming cho hệ thống dự đoán.

3 Hướng Tiếp Cận

3.1 Univariate Time Series Forecasting

Univariate Time Series Forecasting (Dự báo chuỗi thời gian đơn biến – UTSF) là một bài toán dự báo dữ liệu thời gian mà chỉ có một chỉ số quan trọng được xem xét. Điều này có nghĩa là chỉ có một biến độc lập được sử dụng để dự báo giá trị tiếp theo. Ví dụ, trong bài toán dự báo năng lượng tiêu thụ, chỉ có năng lượng là một chỉ số quan trọng được xem xét, trong khi các chỉ số khác về thời tiết như: nhiệt độ, độ ẩm, tốc độ gió, v.v. đều được bỏ qua.

Có nhiều model được sử dụng cho UTSF, bao gồm: ARIMA (AutoRegressive Integrated Moving Average) [1], SARIMA (Seasonal ARIMA) [2], Prophet (Facebook's Time Series Forecasting Library) [3], Neural Network Models (LSTM, GRU, Feedforward, ...), BigDL-Chronos Models (TCNForecaster, LSTMForecaster, Seq2SeqForecaster, ...) [5], ... Tùy vào tính toán và mục tiêu, chúng ta có thể chọn một trong số các model trên để phù hợp với nhu cầu của mình.

3.2 Multivariate Time Series Forecasting

Multivariate Time Series Forecasting (Dự báo chuỗi thời gian đa biến – MTSF) là một bài toán dự báo dữ liệu thời gian mà nhiều chỉ số quan trọng được xem xét. Điều này có nghĩa là nhiều biến độc lập được sử dụng để dự báo giá trị tiếp theo. Ví dụ, trong bài toán dự báo năng lượng tiêu thụ, năng lượng cùng với các chỉ số về thời tiết như: nhiệt độ, độ ẩm, tốc độ gió, v.v. đều được xem xét là các chỉ số quan trọng để dự báo năng lượng tiêu thụ tiếp theo. Bài toán dự báo chuỗi thời gian đã biến đổi hỏi sự tương tác giữa các biến được xem xét và yêu

cầu sử dụng các mô hình phức tạp hơn so với bài toán dự báo đơn biến. Từ đó có thể giúp tăng độ chính xác của dự báo.

Có nhiều model có thể sử dụng cho MTSF, bao gồm: VAR (Vector Autoregression) [4], Prophet (Facebook's Time Series Forecasting Library) [3], Neural Network Models (LSTM, GRU, Feedforward, ...), BigDL-Chronos Models (TCNForecaster, LSTMForecaster, Seq2SeqForecaster, ...) [5], ... Tùy vào tính toán và mục tiêu, chúng ta có thể chọn một trong số các model trên để phù hợp với nhu cầu của mình.

3.3 BigDL – Chronos Forecaster

BigDL [6] là một Deep Learning Framework cho Apache Spark, được công bố vào năm 2016 bởi Intel. Nó cho phép chúng ta xây dựng các mô hình Deep Learning trên Apache Spark và sử dụng tài nguyên tính toán phân tán để tăng tốc quá trình huấn luyện và đánh giá mô hình. BigDL cung cấp một API dễ sử dụng cho Python và Scala, cho phép chúng ta dễ dàng tích hợp với các dự án Spark của mình.

Chronos Forecaster là một thư viện trong BigDL, nó cho phép người dùng áp dụng các thuật toán Deep Learning vào dữ liệu chuỗi thời gian cho mục đích dự báo. Mục tiêu của Chronos Forecaster là cung cấp một giải pháp dễ sử dụng và có thể mở rộng để dự báo chuỗi thời gian, tận dụng sức mạnh của Deep Learning và nền tảng Apache Spark.

TCNForecaster là một mô hình dự báo chuỗi thời gian được xây dựng trên nền tảng BigDL. Nó sử dụng mạng neural Temporal Convolutional Network (TCN) để dự báo giá trị trong tương lai. TCN là một neural network sử dụng kiến trúc mạng tích chập (convolutional networks) thay vì mạng hồi quy (recurrent networks). Nó hỗ trợ các trường hợp multi-step và multi-variant. Causal Convolutions cho phép tính toán song song quy mô lớn khiến TCN có thời gian suy luận ít hơn so với mô hình dựa trên RNN như LSTM.

Seq2SeqForecaster trong BigDL là một mô hình dự báo chuỗi thời gian dựa trên mô hình seq2seq (sequence to sequence). Đây là một trong những kiến trúc mạnh mẽ để xử lý dữ liệu chuỗi thời gian sử dụng Deep Learning. Mô hình Seq2SeqForecaster được thiết kế để dự báo các giá trị liên tục trong tương lai dựa trên các giá trị đầu vào trong quá khứ. Điều này có thể giúp cho việc dự báo về các sự kiện trong tương lai như nhiệt độ, giá cả, doanh số và số lượng đơn hàng, ... Tổng quan, Seq2SeqForecaster là một mô hình dự báo chuỗi thời gian mạnh mẽ và linh hoạt, cung cấp năng suất cao và độ chính xác trong việc dự báo giá trị trong tương lai dựa trên dữ liệu thời gian.

LSTMForecaster là một trong những mô hình sử dụng cho bài toán dự báo chuỗi thời gian được định nghĩa trong thư viện BigDL và có thể chạy trên nền tảng Apache Spark. Nó sử dụng một kiến trúc mạng LSTM để dự báo giá trị tại thời điểm trong tương lai bằng cách học từ các giá trị đầu vào trong quá khứ. LSTM là một kiến trúc mạnh mẽ và linh hoạt cho việc xử lý dữ liệu chuỗi thời gian, vì nó có khả năng giữ lại thông tin trong khoảng thời gian dài và cho phép mô hình phân tích các mối quan hệ giữa các giá trị trong quá khứ và tương lai. Tổng quan, LSTMForecaster là một mô hình dự báo chuỗi thời gian mạnh mẽ

và linh hoạt, cung cấp năng suất cao và độ chính xác trong việc dự báo tình hình trong tương lai dựa trên dữ liệu chuỗi thời gian.

3.4 Convolutional Neural Network

Convolutional Neural Network (CNN) [7] là một kiểu mạng neural Deep Learning được sử dụng để phân loại hoặc dự báo dữ liệu hình ảnh. Nó sử dụng một quá trình convolution để tìm các đặc trưng cụ thể trong dữ liệu và đặt nền tảng cho việc phân loại hoặc dự báo. CNN chủ yếu được sử dụng cho các bài toán xử lý hình ảnh, nhưng cũng có thể được sử dụng cho một số bài toán khác. Với dữ liệu time-series, CNN có thể được sử dụng như một phần của mô hình học sâu để phân loại hoặc dự báo dữ liệu. Trong trường hợp này, dữ liệu time-series có thể được chuyển đổi thành một ma trận và được sử dụng như một hình ảnh đầu vào cho CNN. Nó sẽ sử dụng các lớp convolution để tìm các đặc trưng quan trọng trong dữ liệu và sau đó sử dụng các lớp fully connected để tạo ra một dự báo.

3.5 Gated Recurrent Unit

Recurrent Neural Network (RNN) [8] là một loại neural network hồi quy dựa trên việc sử dụng cùng một hàm tính toán để xử lý các đầu vào liên tục. Trong RNN, trạng thái hiện tại được sử dụng để dự báo trạng thái tiếp theo và quá trình này được lặp đi lặp lại cho mỗi đầu vào trong chuỗi.

Gated Recurrent Unit (GRU) [9] là một phiên bản cải tiến của RNN, được sử dụng trong học sâu cho các bài toán như phân loại văn bản, dự đoán chuỗi thời gian và giải quyết các bài toán Natural Language Processing (NLP) khác. GRU có cấu trúc gồm 2 cổng (gate): cổng cập nhật trạng thái (update gate) và cổng xóa trạng thái (reset gate). Các cổng này dùng để quản lý và giữ nguyên thông tin trong một chuỗi và cho phép model tự động chọn việc cập nhật hay loại bỏ thông tin cũ. Cụ thể, update gate sẽ xác định mức độ cập nhật thông tin mới vào hidden state, trong khi reset gate sẽ xác định mức độ loại bỏ thông tin cũ để chuẩn bị cho việc nhập thông tin mới. Sau đó, một giá trị mới của hidden state được tính toán dựa trên giá trị cũ và thông tin mới. Kết quả cuối cùng của GRU là giá trị của hidden state đó, sẽ được dùng để dự đoán kết quả. GRU có thể hoạt động tốt với dữ liệu time-series bằng cách giữ nguyên những thông tin quan trọng và loại bỏ những thông tin không cần thiết. Vì vậy, GRU có thể dự báo các sự kiện trong tương lai dựa trên dữ liệu quá khứ và hiện tại.

3.6 Long Short-Term Memory

LSTM (Long Short-Term Memory) [10] là một loại mô hình RNN được thiết kế để xử lý dữ liệu chuỗi thời gian, trong đó một số thông tin của trạng thái trước được giữ lại để dùng cho các bước tiếp theo. So với RNN, LSTM có khả năng xử lý tốt hơn các bài toán với dữ liệu chuỗi thời gian dài, vì nó có thể giữ lại những thông tin quan trọng và loại bỏ những thông tin không cần thiết. So với GRU, LSTM có cấu trúc phức tạp hơn và yêu cầu nhiều tài nguyên hơn để huấn luyện, nhưng nó có khả năng xử lý tốt hơn một số bài toán NLP với dữ liệu chuỗi thời gian. LSTM có cấu trúc gồm:

- Memory cell: là phần chứa thông tin quan trọng trong thời gian dài.
- Input gate: điều khiển việc cập nhật thông tin vào memory cell.
- Forget gate: điều khiển việc xóa thông tin không cần thiết từ memory cell.
- Output gate: điều khiển việc truyền thông tin từ memory cell ra ngoài mô hình.

Các phần trên hoạt động đồng thời và những giá trị tính toán từ các neural network đều được sử dụng để cập nhật, xóa hoặc truyền thông tin từ memory cell ra ngoài mô hình. Điều này cho phép mô hình LSTM giữ lại những thông tin quan trọng trong thời gian dài và loại bỏ những thông tin không cần thiết.

3.7 Apache Kafka

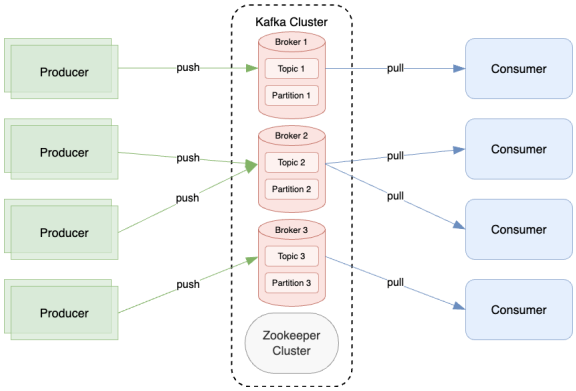
Apache Kafka [11] là một hệ thống mã nguồn mở để xử lý luồng dữ liệu từ các nguồn đầu vào và gửi đến các nguồn đầu ra, cung cấp một cơ chế tự động hóa, mã hóa và gửi lại dữ liệu. Nó cung cấp các tính năng như lưu trữ lịch sử, bảo mật, quản lý bản ghi, tìm kiếm và phân tích. Đặc biệt, Apache Kafka còn cung cấp tính năng mức độ đồng bộ cao, có thể mở rộng dễ dàng, và cho phép phân tán dữ liệu giữa nhiều máy chủ. Apache Kafka có 4 thành phần chính:

- Topics: Là một tập hợp các bản ghi sự kiện (records) được lưu trữ trong một topic. Mỗi topic là một đối tượng riêng biệt để chứa dữ liệu và cho phép cho các hệ thống khác đồng bộ với nó.
- Producers: Là các đối tượng gửi dữ liệu đến các topic. Producers có thể gửi dữ liệu đến nhiều topic cùng một lúc và đảm bảo tính toàn vẹn của dữ liệu truyền tải.
- Brokers: Là các nút trung gian trong một cluster Kafka, chịu trách nhiệm lưu trữ dữ liệu và trao đổi dữ liệu giữa các producers và consumers.
- Consumers: Là các đối tượng nhận dữ liệu từ các topic. Consumers có thể đăng ký với nhiều topic và đồng bộ với chúng để nhận dữ liệu.

Ngoài ra, Apache Kafka còn có một số thành phần khác như: Partitions, Replicas, và Zookeeper, nhưng 4 thành phần chính trên là các thành phần quan trọng nhất để hiểu sơ bộ về Apache Kafka. Kiến trúc của Apache Kafka được thể hiện trong Hình 2 bao gồm các nút broker tách biệt và liên kết với nhau, Producers gửi dữ liệu đến Topics, Consumers đọc dữ liệu từ Topics, và Zookeeper quản lý toàn bộ hệ thống.

3.8 MongoDB

MongoDB là một hệ quản trị cơ sở dữ liệu NoSQL, cho phép lưu trữ và truy xuất dữ liệu dạng Document. Nó hỗ trợ tính năng phân tán và tự động sharding, giúp tăng hiệu suất và khả năng mở rộng dữ liệu. MongoDB là một trong những hệ quản trị cơ sở dữ liệu phổ biến và được sử dụng rộng rãi trong các ứng dụng web và mobile. Có rất nhiều ưu điểm của MongoDB như: *Dữ liệu dạng Document*: MongoDB lưu trữ dữ liệu dạng document, giúp tăng tính linh hoạt



Hình 2: Kafka's communication system.

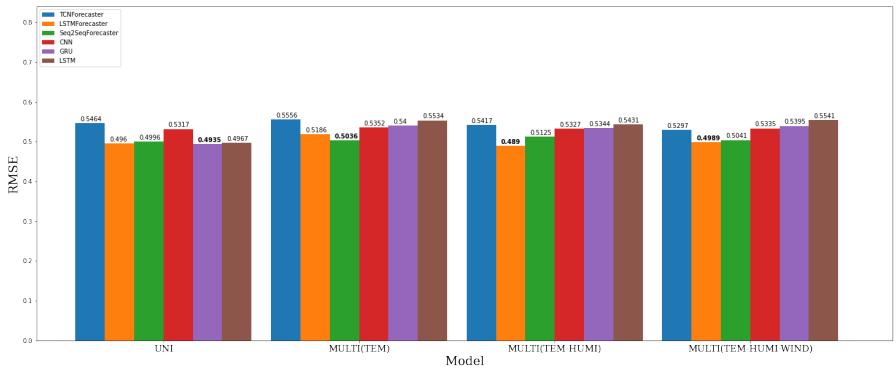
và dễ dàng mở rộng hơn so với các hệ quản trị cơ sở dữ liệu dạng bảng; *Phân tán và Sharding tự động*: MongoDB có tính năng phân tán và sharding tự động, giúp tăng hiệu suất và khả năng mở rộng dữ liệu một cách dễ dàng; *Tính năng Map-Reduce*: MongoDB cung cấp tính năng Map-Reduce, giúp cho việc thống kê và phân tích dữ liệu trở nên dễ dàng hơn; *Tính năng Replication và failover*: MongoDB có hệ thống quản lý bản sao (replication) và tự động failover, giúp tăng tin cậy hệ thống và khả năng dự phòng; *Tương thích với nhiều ngôn ngữ lập trình*: MongoDB tương thích với nhiều ngôn ngữ lập trình như Java, Python, Ruby, v.v., giúp cho việc sử dụng và mở rộng hệ thống trở nên dễ dàng hơn.

Vì Bigdata có khối lượng rất lớn nên việc lưu trữ và quản lý dữ liệu trở nên khó khăn, nhờ vào các tính năng vượt trội của MongoDB chúng tôi sử dụng nó để quản lý dữ liệu lớn này. Mục đích của chúng tôi sử dụng MongoDB trong hệ thống này còn vì muốn tận dụng khả năng multi node, khi hai hay nhiều máy đều chứa dữ liệu cần thì có thể truy cập P2P (peer-to-peer) để truy vấn dữ liệu qua lại, từ đó có thể lấy dữ liệu từ nhiều node khác nhau, đó là bản chất của dữ liệu lớn vì nó có khối lượng rất lớn nên khả năng lưu trữ phân tán cao, dữ liệu khó mà tập trung tại 1 node.

4 Thực Nghiệm và Kết Quả

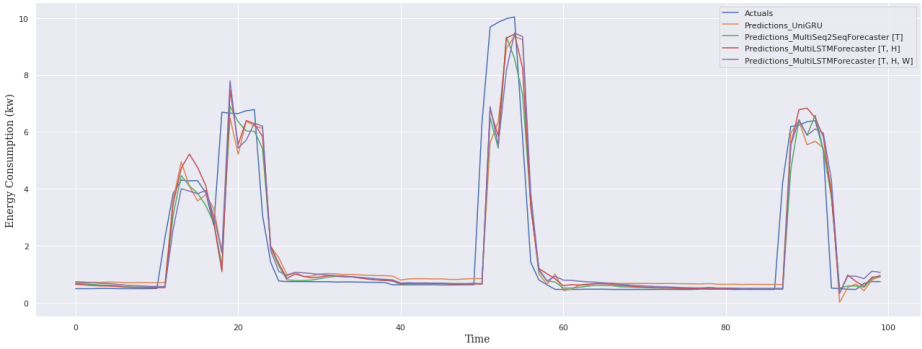
Chúng tôi tiến hành xây các model: CNN, GRU, LSTM, BigDL–Chronos (TCN-Forecaster, LSTMForecaster, Seq2SeqForecaster). Với các model, chúng tôi tiến hành thực nghiệm với các phương pháp UTSF và MTSF. Trong MTSF gồm: MTSF sử dụng biến nhiệt độ (temperature); MTSF sử dụng biến nhiệt độ và độ ẩm (temperature+humidity); MTSF sử dụng biến nhiệt độ, độ ẩm và tốc độ gió (temperature+humidity+windSpeed). Để tránh tính ngẫu nhiên của kết quả thực nghiệm, chúng tôi thực hiện 5 lần chạy và ghi lại kết quả, sau đó tính kết quả trung bình. Hình 3 trình bày kết quả so sánh hiệu suất RMSE trung bình của các model. Với phương pháp UTSF thì mô hình GRU đạt kết quả

tốt nhất theo độ đo RMSE với kết quả $RMSE = 0,4935$. Với phương pháp MTSF sử dụng biến nhiệt độ, mô hình Seq2SeqForecaster cho kết quả tốt nhất với $RMSE = 0,5036$; phương pháp này cho kết quả không tốt bằng phương pháp UTSF ở tất cả các mô hình. Vì vậy, khi thêm biến nhiệt độ vào cũng không cải thiện được hiệu suất của các mô hình. Với phương pháp MTSF sử dụng biến nhiệt độ và độ ẩm, mô hình LSTMForecaster cho kết quả tốt nhất với $RMSE = 0,4890$; đây cũng là kết quả tốt nhất mà chúng tôi thu được với tất cả các phương pháp và mô hình khác nhau. Vì vậy, khi thêm biến nhiệt độ và độ ẩm vào có thể cải thiện được hiệu suất của các mô hình. Với phương pháp MTSF sử dụng biến nhiệt độ, độ ẩm và tốc độ gió, mô hình LSTMForecaster cho kết quả tốt nhất với $RMSE = 0,4989$; phương pháp này cho kết quả không tốt bằng phương pháp UTSF ở đa số các mô hình. Vì vậy, khi thêm biến nhiệt độ, độ ẩm và tốc độ gió vào cũng không cải thiện được đáng kể hiệu suất của các mô hình.



Hình 3: Kết quả trung bình của các model.

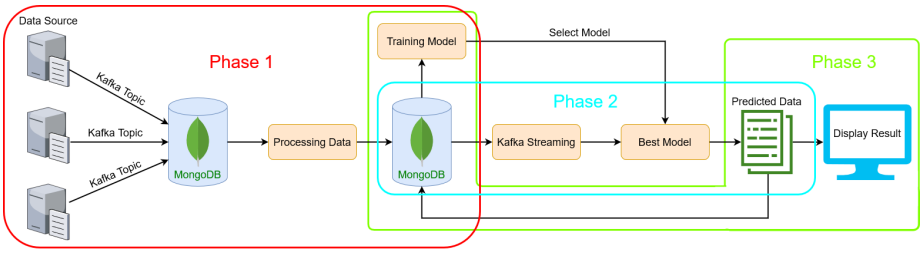
Hình 4 thể hiện kết quả thực nghiệm của các mô hình tốt nhất trên 100 điểm dữ liệu của tập test. Có thể thấy rằng kết quả dự đoán của các mô hình khá tốt, giá trị năng lượng tiêu thụ được dự đoán gần đúng với giá trị năng lượng tiêu thụ thực tế. Bên cạnh đó, kết quả dự đoán của các mô hình khá tương đồng nhau. Dựa vào các kết quả trên, chúng tôi áp dụng mô hình LSTMForecaster với phương pháp MTSF sử dụng biến nhiệt độ và độ ẩm để xây dựng hệ thống trong phần tiếp theo.



Hình 4: Kết quả của các mô hình tốt nhất trên 100 điểm dữ liệu của tập test.

5 Kiến trúc hệ thống

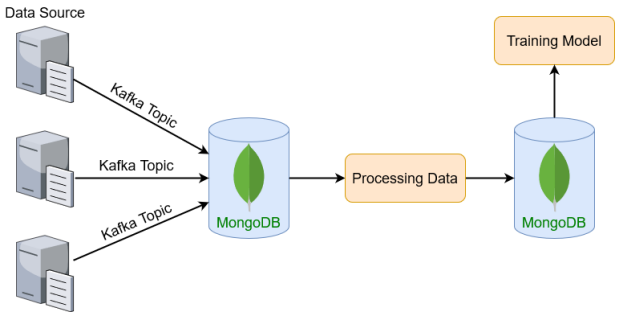
Chúng tôi xây dựng hệ thống với thông tin các Framework được sử dụng như sau: Apache Spark 3.0.1; Apache Kafka 3.2.0 và Scala 2.13; BigDL 2.0. Kiến trúc hệ thống được chia làm ba giai đoạn chính, được thể hiện trong Hình 5.



Hình 5: General architecture of the Energy Consumption Prediction system.

5.1 Giai Đoạn Xây Dựng & Lựa Chọn Mô Hình

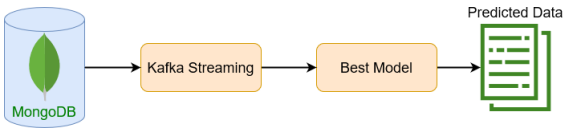
Dữ liệu được tổng hợp từ nhiều nguồn khác nhau sẽ được Producer Kafka gửi vào Topic Kafka. Sau đó Consumer Kafka có vai trò chuyển nhận dữ liệu và truyền dữ liệu trực tiếp vào cơ sở dữ liệu MongoDB để lưu trữ. Dữ liệu thô sau khi được lưu trữ vào MongoDB sẽ được load ra để tiến hành các bước tiền xử lý dữ liệu để thu được một bộ dữ liệu sạch. Sau đó, dữ liệu sẽ được lưu lại thành hai bộ dữ liệu gồm: Traing (dữ liệu dùng để huấn luyện các mô hình) và Streaming (dữ liệu dùng để giả lập hệ thống real-time) vào MongoDB. Tiếp theo, dữ liệu Training được load từ MongoDB để tiến hành huấn luyện các mô hình và chọn ra mô hình tốt nhất để sử dụng làm mô hình chính của hệ thống này. Hình 6 thể hiện đầy đủ chi tiết của giai đoạn này. Các bước xử lý dữ liệu, thực nghiệm



Hình 6: Giai Đoạn Xây Dựng & Lựa Chọn Mô Hình.

huấn luyện mô hình ở Phần 4 cho thấy mô hình LSTMForecaster với phương pháp MTSF sử dụng biến nhiệt độ và độ ẩm đạt kết quả tốt nhất, nên chúng tôi sẽ sử dụng mô hình LSTMForecaster cho hệ thống này.

5.2 Giai Đoạn Stream Dữ Liệu

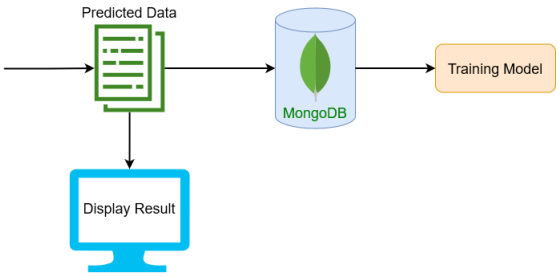


Hình 7: Giai Đoạn Stream Dữ Liệu.

Ở phần này, chúng tôi sử dụng Apache Spark để load dữ liệu Streaming từ MongoDB để giả lập real-time bằng cách dùng Kafka Streaming. Dữ liệu sau khi được load ra sẽ được Producer Kafka gửi vào Topic Kafka, sau đó Consumer Kafka sẽ nhận dữ liệu liên tục từ Producer Kafka để đưa đến mô hình tiến hành quá trình dự đoán. Ở đây chúng tôi sử dụng MongoDB kết hợp với Kafka Streaming để hệ thống cho tốc độ nhận tin và xử lý liên tục. Kafka Streaming giúp xử lý dữ liệu thời gian thực từ nhiều nguồn khác nhau, được sử dụng cho dữ liệu thời gian thực có thể là dữ liệu phi cấu trúc như hình ảnh hoặc văn bản. Hình 7 thể hiện chi tiết giai đoạn giải lập real-time.

5.3 Giai Đoạn Cập Nhật Mô Hình

Dữ liệu được dự đoán xong sẽ được lưu lại vào MongoDB để tiến hành cập nhật lại mô hình sau này. Đồng thời dữ liệu dự đoán cũng sẽ được đưa ra màn hình để người dùng có thể nhìn trực quan kết quả. Hình 8 mô phỏng lại kiến trúc ở giai đoạn này.



Hình 8: Giai Đoạn Cập Nhật Mô Hình.

6 Kết Luận

Trong báo cáo này, chúng tôi đã sử dụng bộ dữ liệu *Smart Home Dataset with Weather Information* với 503,911 điểm dữ liệu để xây dựng hệ thống dự đoán năng lượng tiêu thụ. Chúng tôi đã tiến hành xử lý dữ liệu, lựa chọn phương pháp và tiến hành thực nghiệm để tìm ra kết quả tốt nhất. Từ đó xây dựng nên hệ thống dự đoán năng lượng tiêu thụ real-time để nâng cao sự thoải mái và an ninh cho con người với mức tiêu thụ năng lượng thấp và quản lý năng lượng hiệu quả. Chúng tôi đã áp dụng các phương pháp dự báo chuỗi thời gian đơn biến và đa biến với các mô hình: TCNForecaster, Seq2SeqForecaster, LSTMForecaster, CNN, GRU, LSTM. Chúng tôi sử dụng độ đo RMSE để đánh giá hiệu suất của các mô hình. Kết quả tốt nhất mà chúng tôi đạt được với RMSE là 0.489 sử dụng LSTMForecaster. Bên cạnh các kết quả đạt được thì hệ thống của chúng tôi còn một số hạn chế như kết quả dự đoán mang lại chưa thực sự cao và hệ thống vận hành chưa tốt.

Hướng phát triển trong tương lai:

- Áp dụng thêm nhiều mô hình: AutoformerForecaster, NBeatsForecaster, TCMFForecaster, MTNetForecaster, ARIMAForecaster, ProphetForecaster... để có thể lựa chọn ra phương pháp tốt nhất cho bài toán time series.
- Kết hợp nhiều thuộc tính khác nhau trong bộ dữ liệu để có thể đưa ra kết quả dự đoán chính xác hơn.
- Đưa ra được nhiều kết quả dự đoán cụ thể hơn cho từng loại thiết bị để chúng ta có thể điều chỉnh lượng năng lượng phù hợp.

Tài liệu

1. Shumway, Robert H., David S. Stoffer, Robert H. Shumway, and David S. Stoffer. "ARIMA models." Time Series Analysis and Its Applications: With R Examples (2017): 75-163.
2. Permanasari, Adhistya Erna, Indriana Hidayah, and Isna Alfi Bustoni. "SARIMA (Seasonal ARIMA) implementation on time series to forecast the number of Malaria incidence." In 2013 International Conference on Information Technology and Electrical Engineering (ICITEE), pp. 203-207. IEEE, 2013.

3. Taylor, Sean J., and Benjamin Letham. "Forecasting at scale." The American Statistician 72, no. 1 (2018): 37-45.
4. Stock, James H., and Mark W. Watson. "Vector autoregressions." Journal of Economic perspectives 15, no. 4 (2001): 101-115.
5. Dai, Jason Jinquan, Ding Ding, Dongjie Shi, Shengsheng Huang, Jiao Wang, Xin Qiu, Kai Huang et al. "BigDL 2.0: Seamless Scaling of AI Pipelines from Laptops to Distributed Cluster." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 21439-21446. 2022.
6. Dai, Jason Jinquan, Yiheng Wang, Xin Qiu, Ding Ding, Yao Zhang, Yanzhang Wang, Xianyan Jia et al. "Bigdl: A distributed deep learning framework for big data." In Proceedings of the ACM Symposium on Cloud Computing, pp. 50-60. 2019.
7. Zhao, Bendong, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. "Convolutional neural networks for time series classification." Journal of Systems Engineering and Electronics 28, no. 1 (2017): 162-169.
8. Hüsken, Michael, and Peter Stagge. "Recurrent neural networks for time series classification." Neurocomputing 50 (2003): 223-235.
9. Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014).
10. Hua, Yuxiu, Zhifeng Zhao, Rongpeng Li, Xianfu Chen, Zhiming Liu, and Honggang Zhang. "Deep learning with long short-term memory for time series prediction." IEEE Communications Magazine 57, no. 6 (2019): 114-119.
11. Kreps, Jay, Neha Narkhede, and Jun Rao. "Kafka: A distributed messaging system for log processing." In Proceedings of the NetDB, vol. 11, no. 2011, pp. 1-7. 2011.