Project Report: Word Analogy Finder using Pre-trained Word Embedding Models

Project by team 45

Members : -

Nandan Murari – SE22UARI110

Kalyan – SE22UARI192

Abhi Aitha – SE22UARI009

K Sharath – SE22UARI077

Abstract:

We, focus on the project for pre-trained word embeddings applied particularly on solving word analogy problems: FastText and Word2Vec. Word analogies include finding a word which can complete the analogy such that the semantic relationship for all the words is properly given. We will thereby establish the ability of such models to capture complex relationships and test their performance in an interactive environment.

Introduction

Word embeddings is one of the most fundamental tools for natural language processing (NLP) techniques today, in which words are presented as dense vectors in continuous vector space. These capture several semantic properties of words like similarity, analogy, and contextual relations. Both models adopted for this project, namely Word2Vec and FastText, share the concept of neural networks but handle word representations in different ways.

- Word2Vec (developed by Mikolov et al.) generates word vectors based on surrounding context and has been widely used for word similarity and analogy tasks.
- FastText (developed by Facebook) extends Word2Vec by representing words as bags of character n-grams, which helps improve the handling of out-of-vocabulary (OOV) words, especially rare or morphologically complex words.

This project aims to develop a tool that leverages these pre-trained models to find analogies, thus shedding light on how word embeddings capture semantic relationships between words.

Prior Related Work:

Extensive work has been done on word embeddings. Many tasks such as word similarity and analogy detection have been focused on word embeddings. Word2Vec was one of the earliest models that popularized the concept of word embeddings. Mikolov et al. (2013) presented the Skip-gram and Continuous Bag of Words (CBOW) models, designed to predict a word based on its context or vice versa. These models have proven to capture meaningful relationships, such as analogies: "king" - "man" + "woman" = "queen".

FastText, introduced by Facebook, takes a step forward in representing words not only as whole units but also as sequences of character n-grams. This allows FastText to handle rare or unseen words better than models like Word2Vec, which heavily rely on exact word matches.

Both of the models have been used on tasks such as machine translation, sentiment analysis, and document classification and show their efficiency in capturing the rich semantics of words.

Dataset:

The models employed in this project were pre-trained on large corpora:

▪\tWord2Vec was trained on a humongous dataset of Google News articles with about 100 billion words, and the resulting embeddings represent semantic relationships based on context.

▪FastText was trained on Wikipedia data and includes subword (n-gram) information, allowing it to represent out-of-vocabulary words based on their subword components.

These pre-trained models are available via the Gensim library, making it easy to load and use them for various NLP tasks without needing to train them from scratch.

Methodology:

The methodology for this project can be broken down into several key steps:

1. Model Loading: We use Gensim's API to load pre-trained models such as FastText and Word2Vec. Gensim provides an easy interface to work with these models and allows for efficient similarity computations.

2 fasttext_model = load_model('fasttext-wiki-news-subwords-300')

3 word2vec_model = load_model('word2vec-google-news-300')

4.\tFinding Analogies: The core activity in this assignment is word analogy solving. For an analogy in the form of "A : B :: C :?", we compute the most similar word to C while subtracting the influence of A and adding the influence of B. This can be done using Gensim's most_similar() method that returns words which are most similar in the vector space.

5. Output = model.most_similar(positive=[word_b, word_c], negative=[word_a], topn=1)

For example in the analogy "king:man::queen:? : the function will calculate vector transformation from "king to "man and then, apply that transformation on the "queen" to guess the most likely word to fill the blank.

6.\tTesting the Models: We test the analogy function on a set of sample analogies, such as well-known examples: "king : man :: queen : woman". This is a preliminary validation of the models' performance.

7.\tword_a, word_b, word_c = 'king', 'man', 'woman'

8.\t Interactive Testing: One of the functionalities of the project is the interactive testing of word analogies. Users can enter any three words (A, B, C) and the system returns the word that completes the analogy. This provides room for flexibility and engagement as users explore the relationships among words.

9.\txperiment_ft = find_analogies(fasttext_model, word_a, word_b, word_c)

Experimentations:

The system was tested on many word analogies

1. "king : man :: queen :?"

2. "paris : france :: london :?"

3. "cat : kitten :: dog : ?"

The models were expected to return the correct results:

- Word2Vec and FastText both correctly identified "woman" as the missing word for "king : man :: queen : ?".

- The interactive testing allowed for additional tests, where users could input different analogies and receive the results in real-time.

Results:

The results from testing both models were generally consistent:

-\tFor the analogy "king : man :: queen :?", both Word2Vec and FastText correctly predicted "woman".

-\tSimilarly, for "paris : france :: london :?", both models returned "england".

-\tThe interactive tool allowed the user to test a wide range of analogies and observe the models' predictions.

The results confirm that both Word2Vec and FastText are highly capable of solving word analogies and capturing semantic relationships.

Analysis:

The FastText model performed better than Word2Vec at handling rare or unseen words because it employed subword information. For instance, FastText could discover analogies of uncommon words or words that are rarely spelled, which Word2Vec could not do sometimes.

Both models performed exceptionally well with common words and standard analogies. This points out that pre-trained word embeddings perform very well on typical NLP tasks, especially in word similarity and analogy.

Error Analysis :-

Error analysis is a crucial step in evaluating the performance of the Word Analogy Finder. It helps identify scenarios where the model fails, understand the reasons behind these failures, and suggest possible improvements.

1. **Out-of-Vocabulary (OOV) Words**
   The models struggle with words that are not part of their pre-trained vocabulary. For example, the input analogy `unicorn : mythical :: dragon : ?` may fail because "unicorn" or "dragon" might not exist in the model's vocabulary. FastText handles subwords better than Word2Vec but still has limitations. A possible solution is to notify users about OOV words or switch to more adaptable models.
2. **Context Misinterpretation**
   The models often fail to grasp contextual nuances. For instance, the analogy `bank : money :: river : ?` may lead to incorrect outputs like "finance" instead of "water." This happens because word embeddings do not account for context. While this limitation is inherent to the models used, it can be addressed by highlighting these constraints in the documentation.
3. **Semantic Drift in Analogies**
   The analogy logic may sometimes yield results that do not align with user expectations. For example, the analogy `man : king :: woman : ?` might output "teenage_girl," which reflects biases in the training data. This problem can be mitigated by refining the output through post-processing or acknowledging biases in the project documentation.
4. **Synonym or Polysemy Confusion**
   Words with multiple meanings or synonyms can lead to incorrect results. For instance, in the analogy `bat : animal :: ball : ?`, the model may output "bat" (sports equipment) instead of focusing on the intended meaning of "animal." Using domain-specific embeddings or adding clarification steps for user inputs could help improve performance in such cases.
5. **Performance with Abstract Relationships**
   Abstract relationships tend to produce weaker results. For instance, the analogy `happiness : joy :: sadness : ?` might yield irrelevant outputs due to the models' focus on literal meanings. Limiting the scope of the analogies to more concrete relationships could enhance the reliability of the tool.

To conduct error analysis effectively, we defined metrics like precision of analogy results and user satisfaction scores. By testing various analogies, including both standard and out-of-

context examples, errors were categorized into specific patterns such as OOV issues, contextual misinterpretations, and biases. These findings help identify areas for improvement, such as integrating contextual embeddings like BERT or using domain-specific embeddings.

Future improvements can focus on addressing the root causes of these errors, enhancing the robustness and versatility of the Word Analogy Finder.

Conclusion:

This project shows the strength of FastText and Word2Vec in solving word analogy problems by demonstrating their ability to capture semantic relationships between words. Both models performed well, but with a slight edge for FastText when handling rare words. The interactive analogy testing feature is a great way to keep users engaged with word relationships, making it a very useful tool for educational and research purposes in NLP.

Conclusion Pre-trained word embeddings such as Word2Vec and FastText are really very effective in all the ranges of semantic tasks. These also bring ease to the user while using, making it worth for the developers and researchers in NLP fields.

Real Time uses of this project :-

1. Language Learning and Vocabulary Building

- **Use Case**: Helps learners understand word relationships, synonyms, and antonyms.
- **Example**: For a student learning English, analogies like `happy : joy :: sad : ?` teach connections between emotions and their intensities.

2. Educational Tools

- **Use Case**: Enhances problem-solving and critical thinking skills.
- **Example**: Used in quizzes and puzzles to develop logical reasoning in students (`doctor : hospital :: teacher : ?`).

3. Search Engines and Information Retrieval

- **Use Case**: Improves search engine query handling by understanding the relationship between words.
- **Example**: When a user searches for "biggest river," the system can infer connections and suggest related searches like "Amazon River" or "largest waterways."

4. Chatbots and Virtual Assistants

- **Use Case**: Enhances natural language understanding by interpreting user intent.

- **Example**: If a user asks, "What is similar to a smartphone but larger?" the analogy `smartphone : small :: tablet : large` helps identify "tablet."

5. Recommendation Systems

- **Use Case**: Suggests alternatives based on user preferences by finding similar items.
- **Example**: For a music app, `Taylor Swift : pop :: Billie Eilish : ?` might suggest artists in the same genre.

6. Text Analysis and Summarization

- **Use Case**: Improves algorithms for summarizing text or generating keywords.
- **Example**: Identifying relationships like `important : critical :: minor : trivial` helps refine summaries.

7. Market Research and Trend Analysis

- **Use Case**: Identifies semantic trends in user-generated content.
- **Example**: Analyzing reviews with analogies like `cheap : affordable :: expensive : ?` to understand consumer perceptions.

8. Game Design and Entertainment

- **Use Case**: Creates engaging word puzzle games.
- **Example**: Trivia apps use analogy tasks to entertain and educate users, such as `Batman : Gotham :: Superman : ?`.

9. Creative Writing and Content Generation

- **Use Case**: Inspires ideas by exploring related words and themes.
- **Example**: Finding analogies like `love : heart :: anger : fire` to enrich poetic or narrative content.

10. Cross-Language Translation

Use Case: Enhances the accuracy of translating phrases by mapping relationships.

Example: Understanding `king : queen :: emperor : empress` ensures correct gendered terms in translations.

Contribution of each team member :-

Nandan (SE22UARI-110)-

Using AI softwares and comparing the different codes and finalizing the code and deciding the dataset for the project along with research and referring sources like github and geeksforgeeks, error analysis.

Kalyan (SE22UARI-192)-

Data Collection and preparing the project and research and helping with report writing. Also in data loading and preprocessing .

Aitha Abhi (SE22UARI-009)-

Training the model and deciding which model to use, Building the word analogy function .

K Sharath (SE22UARI-077)-

Research and report writing, testing and evaluation of results.