Recent Advances in Natural Language Processing

Named Entity Recognition & Intent Classification

Eleni Partalidou M.Sc
Software Engineer, M-SENSIS
Sofia Rizou M.Sc, Konstantinos Chatzisavvas Ph.D

msensis

Natural Language Processing
Wikipedia

Natural Language Processing (NLP) is a subfield of *linguistics*, *computer science*, and *artificial intelligence* concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data. The goal is a computer capable of "understanding" the contents of documents, including the contextual nuances of the language within them.
The technology can then accurately extract information and insights contained in the documents as well as categorize and organize the documents themselves.

Challenges in natural language processing frequently involve *speech recognition*, Natural Language Understanding (NLU) and Natural Language Generation (NLG).

Natural Language Processing
Timeline

- **Turing test**
  in Alan Turing's *Computing Machinery and Intelligence* (1950) a criterion of machine intelligence is proposed (what is now called the Turing test). Turing test includes a task that involves the automated interpretation and generation of natural language.

- **Symbolic NLP (1950s – early 1990s)**
  - *Georgetown experiment* (1954) involved fully automatic translation of more than 60 Russian sentences into English
  - SHRDLU / ELIZA: early natural language understanding/processing systems
  - *Conceptual Ontologies* –structured real-world information into computer-understandable data; first chatbots e.g. PERRY

Natural Language Processing
Timeline

- **Statistical / Machine Learning NLP (1990s–2010s)**
  - The computer system gains knowledge by training on diverse data and functioning in that way empirically.
  - After training, the model makes an assumption about a record using statistical rules.

- **Neural Networks / Deep Learning (present)**
  - Neural networks are a category of machine learning models that pass information through layers to perform generalization.
  - Deep Learning is an extension of today's neural networks that concerns using more complex layers to gain knowledge from data.

Natural Language Processing

Use cases

Spam detection

The best spam detection technologies use NLP's text classification capabilities to scan emails for language that often indicates spam or phishing. These indicators can include overuse of financial terms, characteristic bad grammar, threatening language, inappropriate urgency, misspelled company names, and more.

Spam detection is one of a handful of NLP problems that experts consider 'mostly solved' (although you may argue that this doesn’t match your email experience).

Natural Language Processing

Use cases

Machine translation

Truly useful machine translation involves more than replacing words in one language with words of another; it has to capture accurately the meaning and tone of the input language and translate it to text with the same meaning and desired impact in the output language. Machine translation tools are making good progress in terms of accuracy, e.g. Google Translate.

To test any machine translation tool translate text to one language and then back to the original. An oft-cited classic example: Not long ago, translating "*The spirit is willing but the flesh is weak*" from English to Russian and back yielded "*The vodka is good but the meat is rotten.*" Today, the result is "*The spirit desires, but the flesh is weak*," which isn't perfect, but inspires much more confidence in the English-to-Russian translation.

Natural Language Processing

Use cases

Virtual agents and chatbots

Virtual agents such as Apple's Siri and Amazon's Alexa use speech recognition to recognize patterns in voice commands and natural language generation to respond with appropriate action or helpful comments.

Chatbots perform the same magic in response to typed text entries. The best of these also learn to recognize contextual clues about human requests and use them to provide even better responses or options over time. The next enhancement for these applications is question answering, the ability to respond to our questions—anticipated or not—with relevant and helpful answers in their own words.

Natural Language Processing

Use cases

Social media sentiment analysis

NLP has become an essential business tool for uncovering hidden data insights from social media channels. Sentiment analysis can analyze language used in social media posts, responses, reviews, and more to extract attitudes and emotions in response to products, promotions, and events–information companies can use in product designs, advertising campaigns, and more.

Natural Language Processing

Use cases

Text summarization (and generation)

Text summarization uses NLP techniques to digest huge volumes of digital text and create summaries and synopses for indexes, research databases, or busy readers who don't have time to read full text. The best text summarization applications use semantic reasoning and natural language generation (NLG) to add useful context and conclusions to summaries.

Text representation

One hot

In one-hot representation the words of a document are based on a dictionary and are represented as a vector with length the number of words the dictionary has. Each word vector gets the value of 1 at the position in the dictionary and 0 at the rest of the positions.

Text representation

Word2Vec

An unsupervised approach that aims to capture the context of the words in a document by providing a probability distribution for either:

a) a word when the surrounding words are given (Continuous Bag-of-Words learning algorithm, CBOW).

b) A set of surrounding words when a single word is known (Continuous Skip-Gram algorithm, Skipgram).

Text representation

GloVe

Word2Vec models drawback in that they do not take advantage of reputation of words together in a document. GloVe models focus in finding the probability of words being together in a document by creating a frequency table.

Text representation

FastText

FastText models assume that a word is formed by n-grams, where n ranges from 1 to the length of word, and, thus, provides useful embeddings for previously unseen words e.g. out of vocabulary words, languages with large vocabularies and many rare words, morphologically rich languages or typos.

Deep Learning models

LSTM

- Long Short-Term Memory (LSTM) networks are improved versions of Recurrent Neural Networks (RNNs) aiming to capture long-range dependencies in sequential data.
- There are different types of gates in a typical LSTM network:

Forget gate f .: Controls the feedback of the cell state vector ct; decides which information will be preserved and which will be discarded.
Input gate i. : Controls the flow of input signal in the model.
Output gate o. : Controls the flow of cell state vector ct towards the exit.

Deep Learning models

Bidirectional LSTM

- For a given sentence of n words (x_1, x_2, … , x_n) where each word x_i is represented as a d-dimensional vector, an LSTM computes a representation $ht\rightarrow$ of the left-side context of the sentence for every word at timestep t.
- The representation of a word with the use of this model is obtained by the concatenation of its left-side and right-side context representations, $ht = [ht\rightarrow; ht\leftarrow]$.

Deep Learning models

Bidirectional LSTM (CRF Tagging)

- In tasks such as NER there are strong dependencies across output labels since the “grammar” that characterizes interpretable sequences of tags imposes several hard constraints (e.g., I-PER cannot follow B-LOC) that would be impossible to model with independence assumptions.
- Therefore, instead of modelling tagging decisions independently, we model them jointly using a Conditional Random Field (CRF).

Deep Learning models

Transformers

- A novel architecture which aims to solve sequence-to-sequence tasks while handling long range dependencies with ease.
- It adopts the mechanism of attention, differentially weighing the significance of each part of the input data.

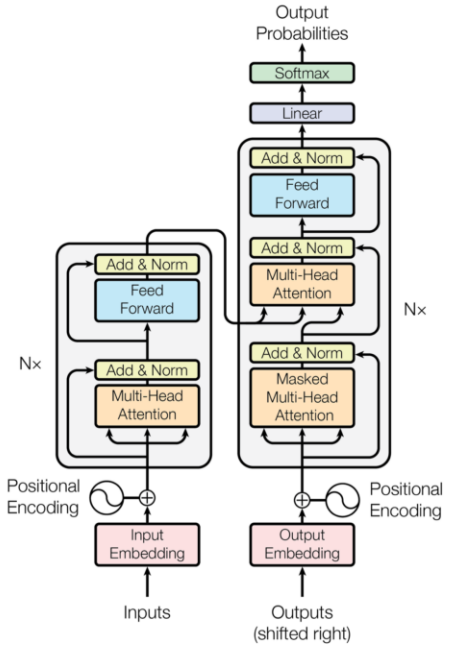


Figure 1: The Transformer - model architecture.

Deep Learning models

BERT

- A deeply bidirectional, unsupervised learning representation pre-trained using only a plain text corpus instead of the vast amount of training data.
- It employs a Transformer architecture that allows massive parallelization during training and produces global dependencies between input and output.
- BERT has already state-of-the-art results in major NLP tasks, such as question answering and Natural Language Inference.

Conversational Agents

- They aim to provide the user with relevant information related to a specific task.
- Some CAs use only text-based input and output, whereas others involve more complex input and output modalities (for instance, speech).
- In conversational agents the goal is to retrieve the intent of a document so the needs of a customer be understood.

Named Entity Recognition

- A subtask of Information Extraction that aims to locate and classify world objects in unstructured text.
- Typical Named Entity Recognition systems include detecting persons, locations and organizations but more complex cases detect dates, numbers or cultural objects.

  Example:
  New York is the most populous city in the United States. ->

  Output:
  New **B** *GPE* | York **I** *GPE* | is **O** | the **O** | most **O** | populous **O** | city **O** |
  in **O**| the **B** *GPE* | United **I** *GPE* | States **I** *GPE*

Intent Classification

- Intent classification uses Machine Learning and Natural Language Processing to automatically associate words or expressions with a particular intent.
- It is useful in computer systems to understand customer needs and gain valuable insights from text.

Examples:

I'm hungry -> **Intent**: *restaurant_search*

Show me good pizza spots

Methodology

LSTM (NER model)

- **Bi-LSTM–CRF:** A bidirectional RNN with long short-term memory units to transform word features into named entity tag scores with CRF. The embedding layer is initialized with pre-trained word embeddings from three different models (Word2Vec, fastText, BERT). Then each word embedding of the sentence passes through a Bi-LSTM unit.

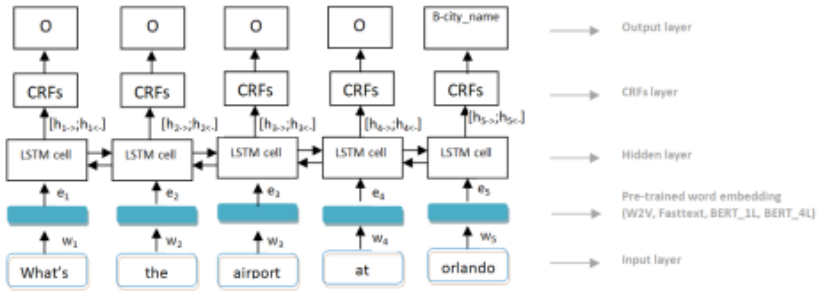


Figure 1: Bi-LSTM–CRF model

Methodology

LSTM (NER model)

- **Bi-LSTM–CRF with CNN character embeddings**: For each word a convolution layer is applied over the characters embeddings of size 3 and then a max pooling layer extracts a new feature vector from the per character feature vectors.

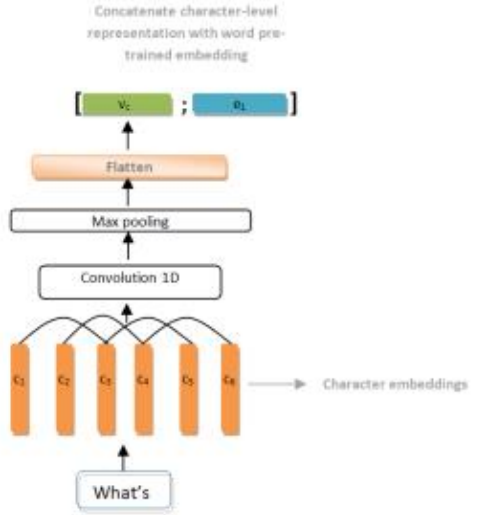


Figure 2: Character embeddings with CNN

Methodology

LSTM (NER model)

- **Bi-LSTM–CRF with LSTM character embeddings**: A character level representation for each word is fed into an LSTM unit and the resulting vector is the last hidden state of the character sequence. Then the resulting vector is concatenated with the pre-trained word embedding.

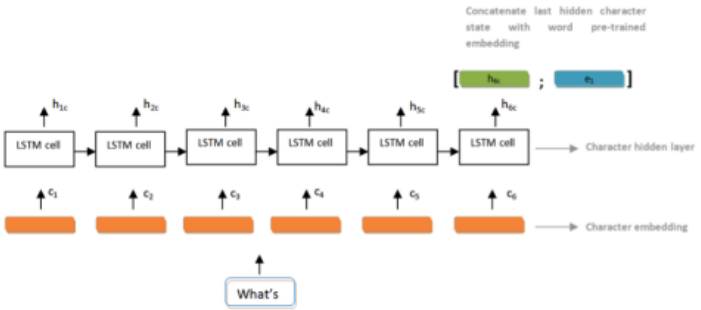


Figure 3: Character embeddings with LSTM

Methodology

LSTM (IC model)

- An SVM classifier is used and the different word embeddings models are compared at the initialization. In an M-class classification context this method processes M binary problems: each one separates one class from the remaining (M – 1) ones.
- The average of the word embeddings of the sentence are used to form its representation.

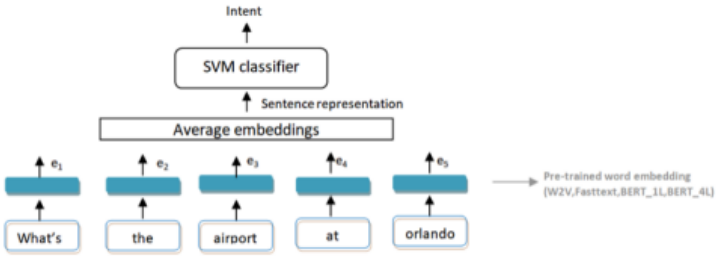


Figure 4: SVM intent classifier

Methodology

LSTM (Unified model)

- In our model we employ an LSTM unit instead of a Gated Recurrent Unit (GRU) compared to previous approaches. The network consists of two kinds of outputs, i.e. the predicted entity label sequence and predicted intent label.
- The entity labels are predicted by the CRFs layer as in the previous models. For the intent prediction, the softmax function is applied to the representation with linear transformation to give the probability distribution over the intent labels.

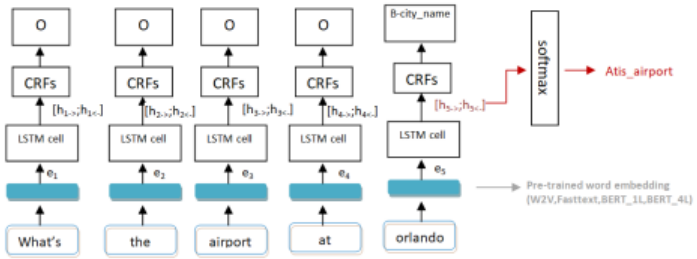


Figure 5: Unified model

Methodology

BERT for Joint IC and NER

- The pre-trained BERT model provides a powerful context-dependent sentence representation and can easily be extended to a joint IC and NER model.
- The intent is predicted as $y^i = \text{softmax}(\boldsymbol{W}^i h_1 + b^i)$ where h_1 is the hidden state of the first special token (CLS).
- For NER, the final hidden states of other tokens h_2 , ..., h_T are feeded into a softmax layer to classify over the labels $y^s{}_n = \text{softmax}(\boldsymbol{W}^s h_n + b^s)$, $n \in 1, ..., N$ where h_n is the hidden state of the first subtoken of the word x_n.

Methodology

Co-Interactive Transformer for Joint NER and IC

- The architecture incorporates a Bi-LSTM with 2 layers as a shared encoder.
- For an input sequence of n tokens $\{x_1, x_2, \dots, x_n\}$ the Bi-LSTM reads it forwardly and backwardly to produce a series of context-sensitive hidden states $H = \{h_1, h_2, \dots, h_n\}$ by repeatedly applying the recurrence $h_i = \text{BiLSTM}(\phi^{\text{emb}}(x_i)h_{i-1})$, where ϕ^{emb} represents the embedding function.
- Firstly, the model performs label attention over intent and entity labels to get explicit representations. Then, they are fed into the co-interactive layer to establish the dependency between the two.

References

Multilingual Name Entity Recognition and Intent Classification employing Deep Learning architectures

S. Rizou, *A. Paflioti*, *A. Theofilatos*, *A. Vakali*, *G. Sarigiannidis*, *K.Ch. Chatzisavvas*.

Thank you for your attention!