ELMO: Embeddings from Language Models

Contextual Word Representations

Ramaseshan Ramachandran

SENSE DISAMBIGUATION OF THE WORD BANK

Synset('bank.n.01')	sloping land (especially the slope beside a body of water)
Synset('depository-financial-institution.n.01')	a financial institution that accepts deposits and channels the money into lending activities
Synset('bank.n.03')	a long ridge or pile
Synset('bank.n.10')	a flight maneuver; aircraft tips laterally about its longitudinal axis (especially in turning)
Synset('bank.v.02')	enclose with a bank
Synset('bank.v.03')	do business with a bank or keep an account at a bank
Synset('bank.v.04')	act as the banker in a game or in gambling
Synset('bank.v.05')	be in the banking business
Synset('deposit.v.02')	put into a bank account
Synset('trust.v.01')	have confidence or faith in

SENSE DISAMBIGUATION - THE WORD PROGRAM

Synset('plan.n.01')	a series of steps to be carried out or goals to be accomplished
Synset('program.n.02')	a system of projects or services intended to meet a public need
Synset('broadcast.n.02')	a radio or television show
Synset('platform.n.02')	a document stating the aims and principles of a political party
Synset('program.n.05')	an announcement of the events that will occur as part of a theatrical or sporting event
Synset('course_of_study.n.01')	an integrated course of academic studies
Synset('program.n.07')	(computer science) a sequence of instructions that a computer can interpret and execute
Synset('program.n.08')	a performance (or series of performances) at a public presentation
Synset('program.v.01')	arrange a program of or for
Synset('program.v.02')	write a computer program

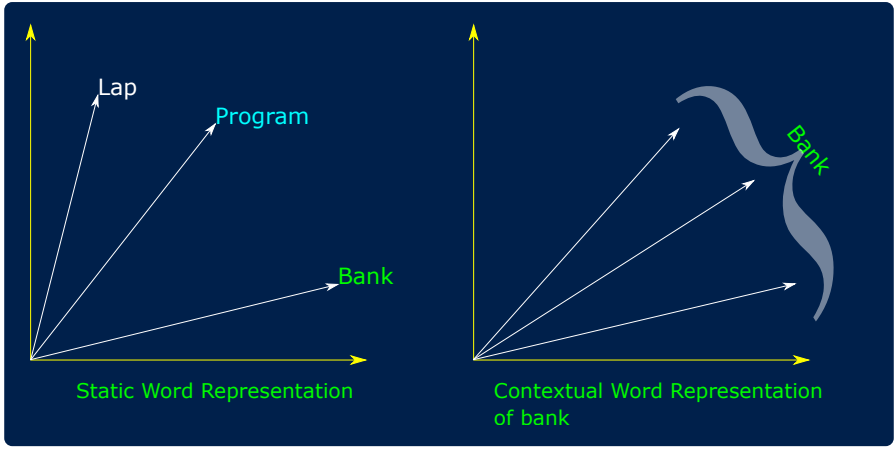
STATIC WORD REPRESENTATION

- Static representation of words
- Irrespective of the context, a polysemous word has the same vector representation
- Insensitive to the context in which they appear
- The word representation of a polysemous word is a biased representation due to certain context/pattern that appear more than any other context in the given corpus
- **It represents syntactic and semantic representations**
- **It does not represent the same word appearing in different linguistic contexts**

LATENT RELATIONSHIP

- The pairs of words that co-occur in similar patterns tend to have semantic relations. Using this property every word in the vocabulary is encoded as dense vectors using vector space semantic models.
- The individual dimensions of the dense vectors are no longer interpretable; a dimension's value merely reflects the strength of the relationship the word has with the rest of the vocabulary
- Probabilistic language models use only nearby contexts; they do not capture relationships across the surrounding sentences of a corpus
- Is it possible to compute the contextual relationship among words, sentences, and paragraphs?

CONTEXTUAL WORD REPRESENTATION



CONTEXTUAL WORD REPRESENTATION

Boys are playing cricket near the Indian **bank**

A fundamental aircraft motion is a **banking** turn

Boys are playing cricket on Cauvery river **bank**

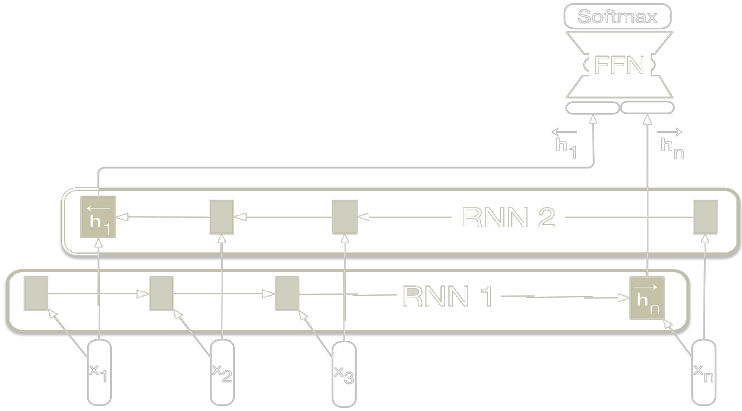
What is your **program** today?

Have you completed the **programming** assignment today?

A cultural **program** was telecast on TV

- The hidden states of an RNN (or the layers of a transformer) hold context-specific representations of the words
- Why not use them to generate contextual word vectors?
- Can we use them for downstream NLP tasks? (This is exactly what ELMo, BERT, and their successors do)
- Each token receives a representation that depends on the whole sentence: $w_i = f(w_1, w_2, w_3, \ldots, w_n)$, where n is the number of tokens in the sentence

BIDIRECTIONAL RNN WITH SINGLE OUTPUT[1]



TRAINING

- The next token in the sequence gets the largest probability mass
- If there are t words in the sequence $(w_1, w_2, w_3, \ldots, w_t)$, the transducer will have $t+1$ input and output neurons

$$\begin{aligned} p(w_{j+1}|w_1, w_2 \ldots, w_j) &= f(\text{RNN}(\theta, w_1, w_2, \ldots, w_j)), \text{ where } j \in [1, t] \\ &= f(\text{RNN}(\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_j)) \end{aligned}$$

and $\hat{w}_j = p(w_j|\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_{j-1})$

- It is possible to create the next token using a conditional context, such as a topic or class of the current content, active or passive voice, etc.
- Example sentence generated without a context
  - *determine the time it takes a piece of glass to hit the ground?* ***A car drives straight off the edge of a cliff***
- Example sentence with context
  - *determine the time it takes the ball to hit the ground,* ***if it is dropped from a height of 45m***.

SEQ2SEQ-BASED APPLICATION

- **Next word/sentence prediction**

  Train on pairs of consecutive sentences to generate context-sensitive sentences in a sequence

- **Question-Answering**

  Given a question (including word problems), find context sensitive answers

- **Auto-encoding**

  Building sentence vectors for identifying similar sentences - extending the distributional hypothesis from words to sentences

- **Machine Translation**

  Given pairs of parallel texts for training, translate an unseen sentence using the trained model

- **Summarization**

  Given a long paragraph, compress it into one or two sentences

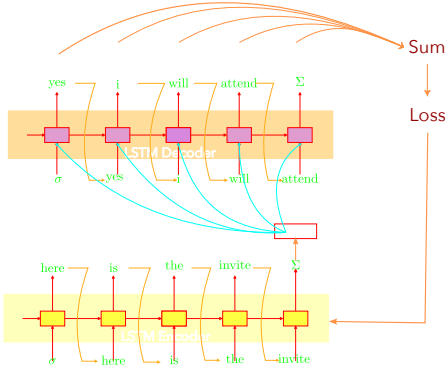
- **Title generation**

  Given abstract and title pairs, train and generate title for any new abstract

- **Chatbots**
- **Discourse Analysis**

  Analysis of written, spoken, sign language

SEQUENCE TO SEQUENCE GENERATOR



ELMO

1. ELMo produces *contextual* word vectors: a word's vector depends on the whole sentence it appears in [2], so $w_i = f(w_1, w_2, w_3, \ldots, w_T)$

$$\text{FORWARD}: \quad p(w_1, w_2, \ldots, w_T) = \prod_k^T p(w_k | w_1, w_2, \ldots\ldots, w_{t-1})$$

$$\text{BACKWARD}: \quad p(w_1, w_2, \ldots, w_T) = \prod_k^T p(w_k | w_{k+1}, w_{k+2}, \ldots\ldots, w_T)$$

2. If there are L layers in both forward and backward LSTMs, then the after L layers, $\overrightarrow{h}_{k,L}$ and $\overleftarrow{h}_{k,L}$ are computed
3. $\overrightarrow{h}_{k,L}$ and $\overleftarrow{h}_{k,L}$ are scaled (may use a layer norm)
4. $\overrightarrow{h}_{k,L}$ and $\overleftarrow{h}_{k,L}$ are concatenated for downstream applications
5. For each token, ELMo takes a task-specific weighted (linear) combination of its representations across the L layers
6. ELMo uses a character-based input layer, so there is no need for an **unknown-token** representation
7. Contrast: ELMo uses stacked biLSTMs; BERT and later models replaced these

WHAT IS LAYER NORMALIZATION?

- Layer Normalization (LayerNorm) is a technique used to normalize the inputs across the features for each training example.
- It helps in stabilizing the learning process in deep neural networks.

Given a vector $x = [x_1, x_2, \ldots, x_d]$, the layer norm is given by

$$\text{LayerNorm}(x) = \gamma \cdot \frac{x - \mu}{\sigma} + \beta \tag{1}$$

where μ is the mean of x, σ is the standard deviation of x and γ and β are learnable scale and shift parameters, respectively

STEPS OF LAYER NORMALIZATION

1. Compute the mean μ of the input vector x. $\mu = \frac{1}{d}\sum_{i=1}^{d} x_i$
2. Compute the variance σ^2 of the input vector x. $\sigma^2 = \frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2$
3. Normalize the input by subtracting the mean and dividing by the square root of variance. $\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$ where ϵ is a small constant for numerical stability.
4. Scale and shift the normalized values. $y_i = \gamma \hat{x}_i + \beta$

LAYERNORM VS BATCHNORM

- **Batch normalization** normalizes each feature across the *batch* dimension. It depends on batch statistics and struggles with variable-length sequences and small batches.
- **Layer normalization** normalizes across the *features* of a single example. It is independent of batch size, which suits sequence models.
- This is why transformers use LayerNorm (or RMSNorm) rather than BatchNorm.

EMBEDDINGS FROM LANGUAGE MODELS (ELMO)

- ELMo is a deep contextualized word representation model.
- Proposed by Peters et al.[2], ELMo captures complex syntactic and semantic characteristics of words.
- ELMo embeddings are context-sensitive, varying for each instance of a word depending on its context.

ELMO ARCHITECTURE OVERVIEW

- ELMo uses a bidirectional LSTM language model with two layers to produce context-sensitive embeddings.
- The model is pretrained on a large text corpus in an unsupervised manner.
- Key components:
  - **Character-based Embeddings**: Word embeddings are computed from character-level CNNs.
  - **Bidirectional LSTMs**: Separate LSTMs for forward and backward contexts.
  - **Layer-wise Aggregation**: Different layers are combined for task-specific representations.

CHARACTER-BASED EMBEDDINGS

- Character-based embeddings allow ELMo to handle out-of-vocabulary (OOV) words and morphological variations.
- A character-level CNN processes each word as a sequence of characters to produce a word embedding.
- This embedding captures both morphological and orthographic information.

$$w = \text{CNN}_{\text{char}}(c_1, c_2, \ldots, c_n)$$

where $c_1, c_2, \ldots, c_n$ are the characters of the word.

BIDIRECTIONAL LANGUAGE MODEL (BILM)

- ELMo uses a bidirectional language model (BiLM) consisting of:
  - **Forward LSTM**: Processes words from left to right.
  - **Backward LSTM**: Processes words from right to left.
- Each LSTM layer l produces hidden states for each token t:

$$\overrightarrow{h}_t^{(l)} = \text{LSTM}_{\text{forward}}(w_{1:t})$$

$$\overleftarrow{h}_t^{(l)} = \text{LSTM}_{\text{backward}}(w_{t:T})$$

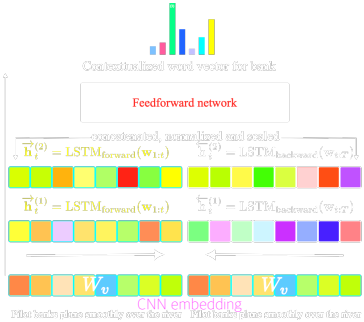
CONTEXTUALIZED ELMO EMBEDDINGS

ELMO embeddings are a weighted combination of hidden states from different layers:

$$\mathrm{ELMO}_t = \gamma \sum_{l=0}^{L} s^{(l)} \cdot \left[\overrightarrow{h}_t^{(l)}; \overleftarrow{h}_t^{(l)} \right]$$

where $s^{(l)}$ are Layer-wise normalized weights using softmax, learned as task-specific parameters, γ is the scaling parameter to adjust the overall magnitude and $\left[\overrightarrow{h}_t^{(l)}; \overleftarrow{h}_t^{(l)} \right]$ are the concatenated hidden states from forward and backward LSTMs.

ELMO ILLUSTRATION



Component	Description
Input	Raw tokens
Layer 1	Input Embedding
Layer 2	Forward
Layer 3	Backward
Output	Contextualized word embedding

$$\text{ELMO}_t = \gamma \sum_{l=0}^{L} s^{(l)} \cdot \left[\overrightarrow{\mathbf{h}}_t^{(l)}; \overleftarrow{\mathbf{h}}_t^{(l)}\right]$$

where $s^{(l)}$ are Layer-wise normalized weights using softmax, learned as task-specific parameters, γ is the scaling parameter to adjust the overall magnitude and $\left[\overrightarrow{\mathbf{h}}_t^{(l)}; \overleftarrow{\mathbf{h}}_t^{(l)}\right]$ are the concatenated hidden states from forward and backward LSTMs.

Pilot banks plane to smoothly over the river
The forward LSTM processes "bank" in the context of "Pilot," while the backward LSTM processes it in the context of "plane smoothly over the river." This dual context helps disambiguate "bank" as a plane maneuver rather than a financial institution

Reference: Deep Contextualized Word Representations

TRAINING THE ELMO MODEL

- ELMO is pretrained as a language model on a large text corpus
- The loss is computed as the sum of the forward and backward language model losses

$$\mathbb{L}(\theta) = \sum_{t=1}^{T} (-\log p(w_t|w_{1:t-1}) - \log p(w_t|w_{t+1:T}))$$

  This includes the parameters of the feed forward weights
- After pretraining, the model is fine-tuned by learning task-specific weights $s^{(l)}$ for each layer.

ELMO INFERENCING I

Feature Extraction: The pre-trained Bidirectional Language Model (biLM) is used as a fixed function to generate high-quality, context-sensitive word vectors for a given input sentence $S = (t_1, t_2, \ldots, t_N)$.

1. **Input Processing**
   - The input token t_k is first converted to an initial context-independent embedding x_k using a **Character Convolutional Neural Network (CNN)**.
2. **Layer-wise Contextual Encoding**
   - The embeddings x_k are passed through the $L = 2$ layers of the pre-trained biLM.
   - **Forward Pass ($\overrightarrow{LM}$):** A forward LSTM computes $\overrightarrow{h}_{k,j}$ (context from $t_1 \ldots t_{k-1}$) for each layer $j \in \{1, 2\}$.
   - **Backward Pass ($\overleftarrow{LM}$):** A backward LSTM computes $\overleftarrow{h}_{k,j}$ (context from $t_{k+1} \ldots t_N$) for each layer $j \in \{1, 2\}$.
   - The total representation at layer j for token t_k is the concatenation of the forward and backward states:

$$R_{k,j} = [\overrightarrow{h}_{k,j}; \overleftarrow{h}_{k,j}]$$

3. **Final Contextual Embedding Calculation:**
   - ELMo defines a set of $L = 3$ layer representations for each token t_k:

$$R_k = \{R_{k,j} \mid j = 0, \ldots, L\}, \quad \text{where } R_{k,0} \equiv x_k \text{ (initial embedding)}$$

   - The final ELMo vector ELMo_k for token t_k is a **task-specific weighted linear combination** of these layer representations:

$$\mathrm{ELMo}_k = E(R_k; \Theta^{\text{task}}) = \gamma^{\text{task}} \sum_{j=0}^{L} s_j^{\text{task}} R_{k,j}$$

4. **Downstream Integration (Inference):**
   - The resulting vector ELMo_k is **concatenated** with the other inputs of the downstream NLP model (e.g., a classifier) and used for the final task prediction.
   - The parameters s_j^{task} (softmax-normalized weights) and γ^{task} (scalar scaling factor) are the **only parameters learned** during the supervised training phase for the specific downstream task; the biLM weights are frozen.

BENEFITS OF ELMO

- ELMo is a task specific combination of the intermediate layer representations in the biLM[2]
- ELMo embeddings are context-sensitive, making them highly effective for tasks requiring nuanced word representations
- They capture complex syntactic and semantic relationships that static embeddings (e.g., Word2Vec, GloVe) cannot
- ELMo is designed to generalize across multiple NLP tasks, providing task-agnostic pre-trained embeddings

CONCLUSION

- ELMo represents a significant advancement in word embeddings by introducing context-dependent representations
- Its architecture, based on character-level embeddings and bidirectional LSTMs, provides a flexible and powerful approach for a range of NLP tasks
- ELMo set the stage for further innovations in contextual embeddings, leading to models like BERT and GPT

INTRODUCTION TO NLP TASKS

- Natural Language Processing (NLP) includes various challenging tasks that require advanced techniques and models
- This presentation covers six critical NLP tasks and explores models that perform well on each task

QUESTION ANSWERING

- **Task**: Answering questions based on provided context or general knowledge.
- **Example**:
  - **Context**: "The cat sat on the mat."
  - **Question**: "Where did the cat sit?"
  - **Answer**: "On the mat."
- **Models**: Bidirectional Encoder Representations from Transformers (BERT), Bidirectional Attention Flow (BiDAF), and today, instruction-tuned LLMs (GPT-4/5, Claude, Gemini, Llama) via prompting

TEXTUAL ENTAILMENT

- **Task**: Determine if one piece of text (hypothesis) can be inferred from another (premise)
- **Example**:
  - **Premise**: "All birds can fly."
  - **Hypothesis**: "Penguins can fly."
  - **Label**: Contradiction. The hypothesis is not entailed (penguins are birds but cannot fly)
- **Models**: Decomposable Attention Model, ESIM (Enhanced Sequential Inference Model), BERT and its successors

SENTIMENT ANALYSIS

- **Task**: Classify text based on its sentiment (positive, negative, neutral).
- **Example**:
  - **Text**: "The movie was fantastic!"
  - **Sentiment**: Positive
- **Models**: LSTM, BERT, CNN for text, Attention-based models.

SEMANTIC ROLE LABELING

- **Task**: Identify and classify the arguments (participants) of a predicate in a sentence
- **Example**:
  - **Sentence**: "John broke the vase."
  - **Predicate**: "broke"
  - **Arguments**:
    - **Agent (who did it)**: John
    - **Patient (what was affected)**: the vase
  - **Models**: RNN-based models, BiLSTM, and Transformer models with SRL-specific adaptations

NAMED ENTITY EXTRACTION

- **Task**: Identify and classify named entities (e.g., person, organization, location) in text
- **Example**:
  - **Text**: "Elon Musk founded SpaceX."
  - **Entities**:
    - **Person**: Elon Musk
    - **Organization**: SpaceX
  - **Models**: BiLSTM-CRF, BERT for NER, and CNN-BiLSTM-CRF

COREFERENCE RESOLUTION

- **Task**: Identify all expressions that refer to the same entity in a text
- **Example**:
  - **Text**: "Mary went to the store. She bought some milk."
  - **Coreference**: "Mary" and "She" refer to the same person

SUMMARY OF TASKS

- These tasks represent key challenges in natural language understanding
- Each task requires different techniques and models, often combining rule-based, statistical, and deep learning approaches.
- Transformer-based architectures have significantly improved performance on all of these tasks
- A single large language model can now perform all six tasks through prompting alone. No task-specific architecture is needed
- Importantly, these tasks survive as *benchmarks*: they are used to evaluate word embeddings and large language models (e.g., in suites such as GLUE and SuperGLUE)

REFERENCES I

[1] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd. Online manuscript released August 24, 2025. 2025. URL: https://web.stanford.edu/~jurafsky/slp3/.

[2] Matthew E. Peters et al. "Deep contextualized word representations". In: *CoRR* abs/1802.05365 (2018). arXiv: 1802.05365. URL: http://arxiv.org/abs/1802.05365.