Modeling Recipe Steps Using Skip-thought Vectors

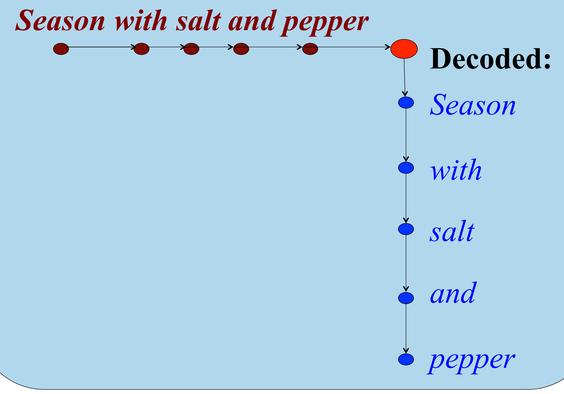
Nicholas Hynes, Mandy Korpusik

Introduction

- Generation of sequences of instructions remains an open problem in NLP.
- Hard to represent steps such that they allow models to do reordering/inference.
- **Goal**: model recipe instructions using skip-thought vectors to allow generation of new recipes.
- We applied skip-thought encodings of recipes to two tasks:
  1. Find the position of a removed step.
  2. Determine which step comes next.

Skip-thoughts

- Word2vec, but on the sentence level.
- Trained using an unsupervised encoder/decoder pair:
  - **Encoder:** GRU (RNN) hidden state represents a sentence as a vector. Training objective maximizes log-probability of previous/next sentences.
  - **Decoder:** GRU unravels encoded vector into original sentence.



Recipe Encodings

- Skip-thoughts encode the given step very well:
  - Original: *Cover and simmer for 1 hour .*
  - Predicted: *Cover and simmer for 1 hour .*
  - Original: *Serve with **steamed Thai** rice .*
  - Predicted: *Serve with **ample steamed** rice .*
- But they cannot predict the next step well:
  - Current step: *Bake for 8-10 minutes or until the edges just begin to brown .*
  - Predicted next step: *Bake until golden brown , about 15 minutes .*

Data

- We scraped 148,000 recipes from websites.

Models

- Baselines:
  - Bag-of-words
  - Mean/median/random encodings
- Proposed:
  - Linear regression using skip-thought encodings as features

Metrics

- Top-1 accuracy: how often the correct location/step was selected.
- Mean rank/median rank: rank of the chosen step in a list of options sorted by closeness.
- BLEU: how well an encoded-decoded step matches the original step.

Task 1: Find Location

- **Goal**: find the location of a missing step from among a sequence of steps in a recipe.

3. Sauté the onions until tender

← ***4. Season with salt and pepper***

5. Remove from heat to cool

- Two approaches (using 5 candidate locations):
  1. **Context**: match next-step prediction of left step to previous-stepprediction of right step.
  2. **Held-ou**t: Match next-step and previous-step predictions to the held-out step.

	Baseline	Context	Held-out
Accuracy	20.7 %	37.4 %	**62.9 %**
Mean rank	3.0	2.3	**1.6**

Task 2: Find Next Step

- **Goal**: determine which candidate recipe step follows the current step.

Season with salt and pepper

?

Remove from heat to cool

Sauté the onions until tender

- Approach: find most similar candidate next step to the predicted next step encoding.

	Baseline	5 steps	10 steps
Accuracy	32.2%	**56.6 %**	42.2 %
Mean rank	2.9	**1.8**	2.8

Extensions

- Similar performance with additional ingredients context vector:
  - Conditional random field (CRF) for tagging ingredient names.
  - Summed ingredient word2vec vectors to get the context vector.
  - Fed the context vector into the skip-thoughts encoder.
- No improvement using LSTM, GRU, or feed-forward neural networks to directly transform the previous step's encoding into the next step's.

Discussion

- Linear model performs fairly well on both tasks, which suggests semantic content in skip-thoughts.
- Predicting the next recipe step is hard, even for neural networks (no better than linear model).

Future Work

- Generating missing steps by selecting from nearest neighbors.
- Training an end-to-end system to predict whether a pair of steps is successive.