CMU CS11-737: Multilingual NLP

Improved Modeling/Learning Methods for Sequence Labeling and Classification

Graham Neubig

Carnegie Mellon University
Language Technologies Institute
Text Classification

- Given an input text $X$, predict an output label $y$

### Topic Classification

<table>
<thead>
<tr>
<th>I like peaches and pears</th>
<th>food</th>
<th>politics</th>
<th>music</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I like peaches and herb</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Language Identification

<table>
<thead>
<tr>
<th>I like peaches and pears</th>
<th>English</th>
<th>Japanese</th>
<th>German</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>桃と梨が好き</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Sentiment Analysis (sentence/document-level)

<table>
<thead>
<tr>
<th>I like peaches and pears</th>
<th>positive</th>
<th>neutral</th>
<th>negative</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I hate peaches and pears</td>
<td>positive</td>
<td>neutral</td>
<td>negative</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

... and many many more!
Sequence Labeling

- Given an input text $X$, predict an output label sequence $Y$ of equal length!

Part of Speech Tagging

He       saw       two       birds
PRON     VERB      NUM       NOUN

Lemmatization

He       saw       two       birds
he       see       two       bird

Morphological Tagging

He       saw       two       birds
PronType=prs  Tense=past,  NumType=card  Number=plur
  VerbForm=fin

... and more!
Reminder: Bi-RNNs

- Simple and standard model for sequence labeling (or classification)

I like these pears

Diagram:

```
<table>
<thead>
<tr>
<th>I</th>
<th>like</th>
<th>these</th>
<th>pears</th>
</tr>
</thead>
<tbody>
<tr>
<td>lookup</td>
<td>lookup</td>
<td>lookup</td>
<td>lookup</td>
</tr>
<tr>
<td>RNN</td>
<td>RNN</td>
<td>RNN</td>
<td>RNN</td>
</tr>
<tr>
<td>RNN</td>
<td>RNN</td>
<td>RNN</td>
<td>RNN</td>
</tr>
<tr>
<td>RNN</td>
<td>RNN</td>
<td>RNN</td>
<td>RNN</td>
</tr>
<tr>
<td>RNN</td>
<td>RNN</td>
<td>RNN</td>
<td>RNN</td>
</tr>
<tr>
<td>concat</td>
<td>concat</td>
<td>concat</td>
<td>concat</td>
</tr>
<tr>
<td>softmax</td>
<td>softmax</td>
<td>softmax</td>
<td>softmax</td>
</tr>
<tr>
<td>PRON</td>
<td>VERB</td>
<td>DET</td>
<td>NOUN</td>
</tr>
</tbody>
</table>
```
Issues w/ Simple BiRNN

- **RNN model** may be insufficient/inefficient
  - Alternative models such as bag-of-ngrams, CNNs (this lecture), or transformers

- **Rare words (esp. w/ rich morphology)** are difficult to learn
  - Character-based word embeddings or subword segmentation

- **Little to no labeled data** exists for many languages
  - Unsupervised learning or cross-lingual transfer

- **Labels depend on each-other**, and predicting independently is not efficient
  - Structured prediction algorithms such as CRFs
Alternative Sequence Models:
Bag-of-ngrams and CNN
(a very brief summary)
Problem w/ RNN: Vanishing Gradient

- Gradients decrease as they get pushed back

\[
\frac{dl}{dh_0} = \text{tiny} \quad \frac{dl}{dh_1} = \text{small} \quad \frac{dl}{dh_2} = \text{med.} \quad \frac{dl}{dh_3} = \text{large}
\]

- Why? “Squashed” by non-linearities or small weights in matrices

- Alternative architectures such as LSTMs/GRUs provide solutions, but still indirect
Alternative: Bag of n-grams

- **Advantage:** Much more direct modeling of short text fragments
- **Disadvantage:** Huge number of parameters for large corpora
1-dimensional Convolutions/Time-delay Networks

I like these pears

\[
\text{tanh}(W^*[x_1;x_2]+b) \quad \text{tanh}(W^*[x_2;x_3]+b) \quad \text{tanh}(W^*[x_3;x_4]+b)
\]

These are soft 2-grams!

probs

- **Advantage**: Can generalize better than bag-of-n-grams, compose multiple layers together

- **Disadvantage**: Less direct than bag of n-grams when remembering individual n-grams important

Modeling Vocabulary Items
Converting Words into Vectors

• Each word, independently, gets its own vector rep.

- Standard lookup entails

  - I
  - like
  - these
  - pears

  lookup
  lookup
  lookup
  lookup

  vector size
  num. words
  lookup(2)
Unknown Words

• Necessity for UNK words
  • We won’t have all the words in the world in training data
  • Larger vocabularies require more memory and computation time

• Common ways:
  • Frequency threshold (usually UNK <= 1)
  • Rank threshold
Solution 1: 

**Fully Character-based Models**

- Basic idea: just encode characters for the entire sentence

![Diagram of character model processing text](chart)

- Issues: slow, ignores natural word boundaries, can be hard to train

Solution 2: Sub-word Segmentation

• Basic idea: separate rarer words into subwords, then embed
  these are some subwords

  segment

  these are some sub_word_s

• **Supervised segmentation:** use supervised word segmentation (e.g. Jieba for Chinese, KyTea for Japanese), or morphological analysis

• **Unsupervised segmentation:** use data driven algorithms to discover which units to segment
Unsupervised Subword Segmentation Algorithms

• **Byte pair encoding:**
  • segment into characters,
  • merge most-frequent subword sequence for fixed number of operations

• **Unigram-based segmentation:**
  • create vocabulary of most-frequent character n-grams
  • use EM algorithm to optimize probabilities, remove subwords with low probability

• Both implemented in "sentencepiece" toolkit
  https://github.com/google/sentencepiece

Sennrich et al. "Neural Machine Translation of Rare Words with Subword Units." ACL 2016.
(Multilingual) Subword Segmentation
Pros/Cons

- **Pros:** very simple (just pre-processing!), fast, somewhat effective

- **Cons:**
  - Cannot handle non-concatenative morphology (e.g. goose -> gee_ se)
  - When trained multilingually, few subwords for less-resourced languages -> over-segmentation

http://juditacs.github.io/2019/02/19/bert-tokenization-stats.html

- Inconsistent segmentation across languages
Solution 3:

Sub-word Based Embeddings

- Basic idea: two-level embedding
  - Characters -> word embeddings
  - Word -> sentence embeddings
Sub-word Based Embedding Models

- BiRNN/LSTM models
  

- CNN models
  

- Bag-of-ngram embedding models
  

- My impression: more direct methods tend to perform better: bag-of-ngram > CNN > BiRNN

- Bag-of-ngrams used in widely-used tools such as fasttext
Embeddings for Cross-lingual Learning: Soft Decoupled Encoding

- Desiderata for cross-lingual lexical learning
  - Spellings are similar (eng "Madrid", spa "Madrid"),
  - with some consistent differences (eng "revolution", spa "revolución"),
  - and some words completely different (eng "hand", spa "mano").

Unsupervised Learning of Representations
Labeled/Unlabeled Data

• **Problem:** we have very little labeled data for most analysis tasks for most languages

• **One Solution:** semi-supervised learning where we also learn from unlabeled data

• Types of semi-supervised learning:
  • Learning of the *neural feature extractor* on a different task (e.g. language modeling)
  • Learning of the *actual output predictor* as well (e.g. form a generative model)

• Former is very easy, latter is more complicated but can be effective
Joint Multi-task Learning

• Train representations to do well on multiple tasks at once

this is an example

\[
\text{Encoder} \rightarrow \begin{array}{c}
\text{LM} \\
\text{Tagging}
\end{array}
\]

• In general, as simple as randomly choosing minibatch from one of multiple tasks
Pre-training

• First train on one task, then train on another

this is an example - Encoder → [dots] → LM

Initialize

this is an example - Encoder → [dots] → Tagging

• Often uses language models, masked language models, or other methods
Unsupervised Objective (1): Language Modeling

• Predict the probability of a sequence, e.g. by predicting next word left to right

\[ P(X) = \prod_{i=1}^{I} P(x_i \mid x_1, \ldots, x_{i-1}) \]

• Like tagging, where tag is the next word!

• Disadvantage: cannot use bidirectional context
Unsupervised Objective (2): Masked Language Modeling

- Denoising auto-encoding is a task of trying to predict clean data from a noised data
  \[ P(X|X') \]

- In NLP, masked language modeling a typical example:

- **Advantage**: can easily use bidirectional context
- **Disadvantage**: is not actually a language model, cannot easily do generation or sequence scoring
Thinking about Multi-tasking, and Pre-trained Representations

- Many methods have names like BERT, mBERT, XLM, XLM-R, Unicoder along with pre-trained models
- These often refer to a combination of
  - **Model:** The underlying neural network architecture
  - **Training Objective:** What objective is used to pre-train
  - **Data:** What data the authors chose to use to train the model
- Remember that these are often conflated (and don't need to be)!
BERT

• **Model:** Multi-layer self-attention. Input sentence or pair, w/ [CLS] token, subword representation

• **Objective:** Masked language modeling + next-sentence prediction

• **Data:** BooksCorpus + English Wikipedia

Other Monolingual BERTs

- BERT has been trained in many languages!
- BERT (en), CamemBERT (fr), BERTje (nl), BETO (es), Ru-BERT (ru), Chinese BERT (zh), BERTurk (tr), and GreekBERT (el)
mBERT

- **Model:** Same as BERT, subword vocabulary created slightly upweighting low-resource languages
- **Objective:** Same as BERT
- **Data:** 100 languages worth of Wikipedia
- **Results:** Quite impressive! (for the time), see below paper for details:

**XLM**

- **Model:** Same as mBERT
- **Objective:** Masked language modeling with parallel sentences:


- **Data:** multilingual Wikipedia (regular MLM), parallel data such as UN/EuroParl/etc. (translation LM)
XLM-R

- **Model:** Same as mBERT
- **Objective:** Same as mBERT (not XLM!)
- **Data:** CommonCrawl + Wikipedia

XTREME: Comparing Multilingual Representations

- 10 tasks, 40 languages w/ different data availability

- Many conclusions, e.g.:
  - Success of pre-training+transfer varies greatly by task/language
  - Different models better at different tasks

Structured Prediction Models
Why Call it “Structured” Prediction?

• Classes are too numerous to enumerate

• Need some sort of method to exploit the problem structure to learn efficiently

• Example of “structure”, the following two outputs are similar:
  
  PRP VBP DT NN  
  PRP VBP VBP NN
Why Model Interactions in Output?

- Consistency is important!

```
time flies like an arrow
NN  VBZ  IN  DT  NN  (time moves similarly to an arrow)
NN  NNS  VB  DT  NN  (“time flies” are fond of arrows)
VB  NNS  IN  DT  NN  (please measure the time of flies similarly to how an arrow would)
↓ max frequency
NN  NNS  IN  DT  NN  (“time flies” that are similar to an arrow)
```
Sequence Labeling w/ BiRNNs

• Still not modeling output structure! Outputs are independent
Recurrent Decoder

I hate this movie <s>
Local Normalization vs. Global Normalization

- **Locally normalized models**: each decision made by the model has a probability that adds to one

\[
P(Y \mid X) = \prod_{j=1}^{\left|Y\right|} \frac{e^{S(y_j \mid X,y_1,\ldots,y_{j-1})}}{\sum_{\tilde{y}_j \in V} e^{S(\tilde{y}_j \mid X,y_1,\ldots,y_{j-1})}}
\]

- **Globally normalized models (a.k.a. energy-based models)**: each sequence has a score, which is not normalized over a particular decision

\[
P(Y \mid X) = \frac{e^{\sum_{j=1}^{\left|Y\right|} S(y_j \mid X,y_1,\ldots,y_{j-1})}}{\sum_{\tilde{Y} \in V_*} e^{\sum_{j=1}^{\left|\tilde{Y}\right|} S(\tilde{y}_j \mid X,\tilde{y}_1,\ldots,\tilde{y}_{j-1})}}
\]
Conditional Random Fields

General form of globally normalized model

\[
P(Y|X) = \frac{\psi(Y, X)}{\sum_Y \psi(Y', X)}
\]

First-order linear CRF

\[
P(Y|X) = \frac{\prod_{i=1}^{L} \psi_i(y_{i-1}, y_i, X)}{\sum_Y \prod_{i=1}^{L} \psi_i(y'_{i-1}, y'_i, X)}
\]
Potential Functions

\[ \psi_i(y_{i-1}, y_i, X) = \exp(W^T T(y_{i-1}, y_i, X, i) + U^T S(y_i, X, i) + b_{y_{i-1}, y_i}) \]
BiLSTM-CRF for Sequence Labeling

CRF Training & Decoding

\[ P(Y|X) = \frac{\prod_{i=1}^{L} \psi_i(y_{i-1}, y_i, x)}{\sum_{Y'} \prod_{i=1}^{L} \psi_i(y'_{i-1}, y'_i, x)} = \frac{\prod_{i=1}^{L} \psi_i(y_{i-1}, y_i, x)}{Z(x)} \]

Go through the output space of Y which grows exponentially with the length of the input sequence using dynamic programming.
Discussion Questions
Discussion

• We have covered three topics in improving sequence models. Read *at least one* of the following papers:
  
  • Multilingual Neural Machine Translation With Soft Decoupled Encoding (Topic: Subword Models)
  
  • Cross-lingual Language Model Pretraining (Topic: Unsupervised Learning of Representations)
  
  • XTREME: A Massively Multilingual Multi-task Benchmark for Evaluating Cross-lingual Generalization (Topic: Unsupervised Learning of Representations)
  
  • Bidirectional LSTM-CRF Models for Sequence Tagging (Topic: Structured Prediction Models)

• For the paper you read, what is one problem with a standard sequence labeling model that the methods presented or discussed therein resolve? What is one problem that you think that the discussed methods would still *not* be able to resolve?
Thank You!