Text Analytics & Business Application

Text Classification 2

Qinglai He
Department of Operations and Information Management
Wisconsin School of Business

WISCONSIN SCHOOL OF BUSINESS

Text Classification (Cont.)

- ~~Hand-coded rules / rule-based classifier~~
- ~~One pipeline, many classifiers~~
  - ~~Naïve bayes~~
  - ~~Logistic regression~~
  - ~~Support vector machine~~
- Use neural embeddings in text classification
  - Word embeddings
  - Subword embeddings and fastText
- Deep learning for text classification
  - LSTMs

Potential Reasons for Poor Classifier Performance

1. Since we extracted all possible features, we ended up in a **large**, **sparse** feature vector, where most features are too rare and end up being noise. A sparse feature set also makes training hard.
2. There are very few examples of relevant articles (~20%) compared to the non-relevant articles (~80%) in the dataset. This class **imbalance** makes the learning process skewed toward the non-relevant articles category, as there are very few examples of "relevant" articles
3. Perhaps we need a better learning **algorithm**
4. Perhaps we need a better **pre-processing** and **feature extraction** mechanism
5. Perhaps we should look to tuning the classifier's **parameters** and hyperparameters.

Approach 3: Use Neural Embedding in Text Classification (1)

Word Embedding

Using Neural Embeddings in Text Classification

- The advantage of using embedding-based features:
  - They create a dense, low-dimensional feature representation instead of the sparse, high-dimensional structure of BoW/TF-IDF and other such features.

Word Embeddings: Coding Examples

- We'll use a **pre-trained** embedding model:
  - Gensim: Word2vec
  - GloVe
- This is a large model that can be seen as a dictionary where the keys are words in the vocabulary and the values are their learned embedding representations.
- Accuracy = 80% (trained with a logistic regression classifier)

```python
# Creating a feature vector by averaging all embeddings for all sentences
def embedding_feats(list_of_lists):
    DIMENSION = 300
    zero_vector = np.zeros(DIMENSION)
    feats = []
    for tokens in list_of_lists:
        feat_for_this =  np.zeros(DIMENSION)
        count_for_this = 0
        for token in tokens:
            if token in w2v_model:
                feat_for_this += w2v_model[token]
                count_for_this +=1
        feats.append(feat_for_this/count_for_this)
    return feats

train_vectors = embedding_feats(texts_processed)
print(len(train_vectors))
```

Note that the above code will give a single vector with DIMENSION(=300) components. We treat the resulting embedding vector as the feature vector that represents the entire text.

Our Own Embeddings VS. Pre-trained Embeddings

When to use?

Compute the vocabulary overlap. If the overlap between the vocabulary of our custom domain and that of pre-trained word embeddings is greater than **80%,** pre-trained word embeddings tend to give good results in text classification.

Note:

- Learned or pre-trained embedding models have to be stored and loaded into memory while using these approaches.
  - If the model itself is bulky (e.g., the pre-trained model we used takes 3.6 GB), we need to factor this into our deployment needs

Approach 4: Deep Learning for Text Mining

Deep Learning for Text Classification

- Deep learning: a family of machine learning algorithms where the learning happens through different kinds of multilayered neural network architectures
- Two of the most commonly used neural network architectures for text classification
    - Convolutional Neural Networks (CNNs)
    - Recurrent Neural Networks (RNNs)
        - Long short-term memory (LSTM) networks are a popular form of RNNs

Steps to Train a DL Model

1. Tokenize the texts and convert them into word index vectors.
2. **Pad** the text sequences so that all text vectors are of the same length.
3. Map every word index to an embedding vector. We do that by multiplying word index vectors with the embedding matrix. The embedding matrix can either be populated using pre-trained embeddings or it can be trained for embeddings on this corpus.
4. Use the output from Step 3 as the input to a neural network architecture.

LSTMs for Text Classification

- Very popular in recent years
  - Language is sequential in nature
  - RNNs are specialized in working with sequential data

- RNNs work on the principle of using this context while learning the language representation or a model of language.
  - Hence, they're known to work well for NLP tasks.

LSTMs: Coding Examples (with training your own embedding)

- This code may take a while to run.
- LSTMs are more powerful in utilizing the sequential nature of text
- Test accuracy with RNN: 0.79

```
print("Defining and training an LSTM model, training embedding layer on the fly")
rnnmodel = Sequential()
rnnmodel.add(Embedding(MAX_NUM_WORDS, 128))

rnnmodel.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
rnnmodel.add(Dense(2, activation='sigmoid'))
rnnmodel.compile(loss='binary_crossentropy',
                 optimizer='adam',
                 metrics=['accuracy'])
print('Training the RNN')
rnnmodel.fit(x_train, y_train,
             batch_size=32,
             epochs=1,
             validation_data=(x_val, y_val))
score, acc = rnnmodel.evaluate(test_data, test_labels,
                               batch_size=32)
print('Test accuracy with RNN:', acc)
```

Why Deep Learning Model Is Not Yet the Silver Bullet for NLP?

- Overfitting on small dataset
- Require domain adaption
- Limited model interpretability
- Cost might be high
- …

Practical Advice

1. Practical Advice: No Training Data

- Let’s say we’re asked to design a classifier for segregating customer complaints for our e-commerce company.
- Classifier: route customer complaint emails into a set of categories: billing, delivery, and others.
- What if a historical database doesn’t exist?
    - where should we start to build our classifier?

1. Practical Advice: No Training Data

Solution 1:

- Creating an **annotated dataset** where customer complaints are mapped to the set of categories mentioned above
- Get customer service agents to **manually label** some of the complaints and use that as the training data for our ML model

Solution 2:

- “Bootstrapping” or “weak supervision.” There can be certain patterns of information in different categories of customer requests
  - Delivery-related requests talk about shipping, delays, etc.

  We can get started with compiling some such patterns and using their presence or absence in a customer request to label it, thereby creating a small (perhaps noisy) annotated dataset for this classification task

2. Practical Advice: Less Training Data

Solution: Active learning

1. Train the classifier with the available amount of data.
2. Start using the classifier to make predictions on new data
3. For the data points where the classifier is very unsure of its predictions, send them to **human annotators** for their correct classification.
4. Include these data points in the existing training data and retrain the model

Tools like Prodigy have active learning solutions implemented for text classification.

3. Practical Advice: A Lot of Training Data

- Models are inherently biased toward the kind of language seen in the training data

- Solution: Domain adaptation(a.k.a., transfer learning). Steps are as below:
  1. Start with a large, **pre-trained language** model trained on a large dataset of the source domain (e.g., Wikipedia data).
  2. **Fine-tune** this model using the target language's unlabeled data.
  3. **Train** a classifier on the **labeled target domain data** by extracting feature representations from the fine-tuned language model from Step 2.

4. Other Practical Advice

- **Establish strong baselines**
  - It's always good to start with simpler approaches and try to establish strong baselines first.
- **Balance training data**
  - An imbalanced dataset can adversely impact the learning of the algorithm and result in a biased classifier
- **Combine models and humans in the loop**
  - In practical scenarios, it makes sense to combine the outputs of multiple classification models with handcrafted rules from domain experts to achieve the best performance for the business.
- **Make it work, make it better**
  - Building a classification system is not just about building a model. For most industrial settings, building a model is often just 5% to 10% of the total project.
- **Use the wisdom of many**
  - Every text classification algorithm has its own strengths and weaknesses. There is no single algorithm that always works well.

Summary of Text Classification

- **Approach 1:** Hand-coded rules / rule-based classifier
- **Approach 2:** One pipeline, many classifiers
  - Naïve bayes
  - Logistic regression
  - Support vector machine
- **Approach 3:** Use neural embeddings in text classification
  - Word embeddings
- **Approach 4:** Deep learning for text classification
  - LSTMs

Approaches to NLP

Approach 1

- Heuristic-based NLP
  - Early attempts at building NLP systems were based on building rules for the task at hand.
  - Such systems normally require resources like dictionaries and thesauruses.
  - E.g., Lexicon-based sentiment analysis, Wordnet, regular expression

Approach 2+3

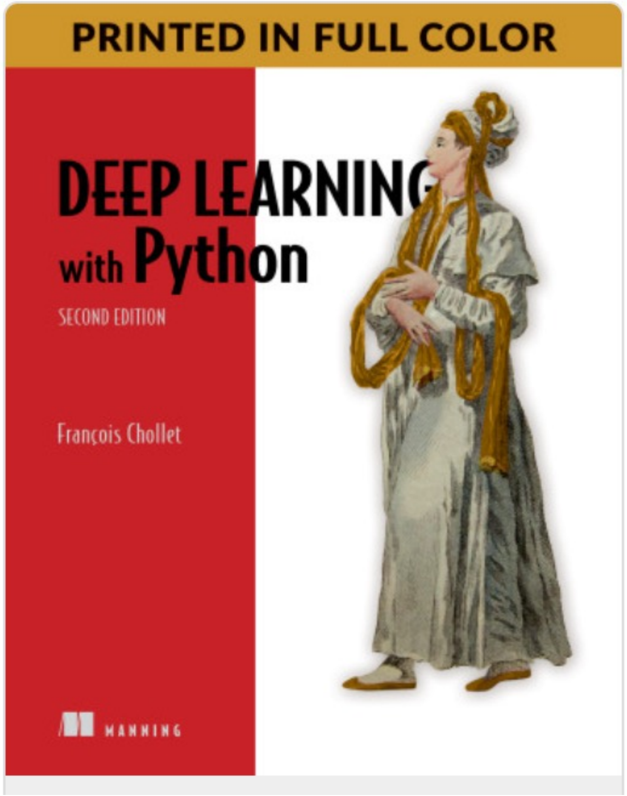
- Machine learning for NLP
  - Naïve baye, support vector machine, hidden Markov model, conditional random fields…

Approach 4+3

- Deep learning for NLP
  - Recurrent neural networks (RNN), long short-term memory (LSTM), convolutional neural networks (CNN), transformers…

Additional Resources

Learn more about deep learning and deep neural network models



Deep Learning with Python, Second Edition

4.9 ★★★★★ 93 customer reviews | 4.7 ★★★★★ 308 amazon ratings

François Chollet

October 2021 · ISBN 9781617296864 · 504 pages

printed in color · includes free previous edition eBook

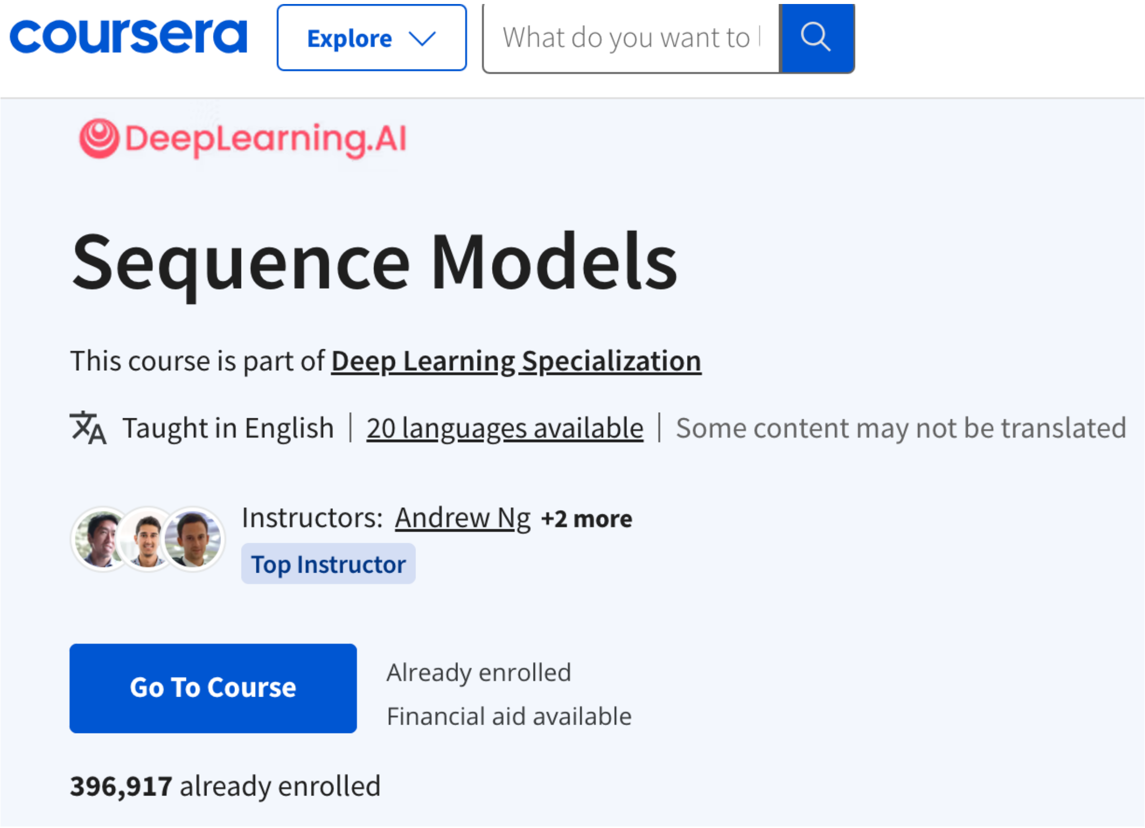
Data

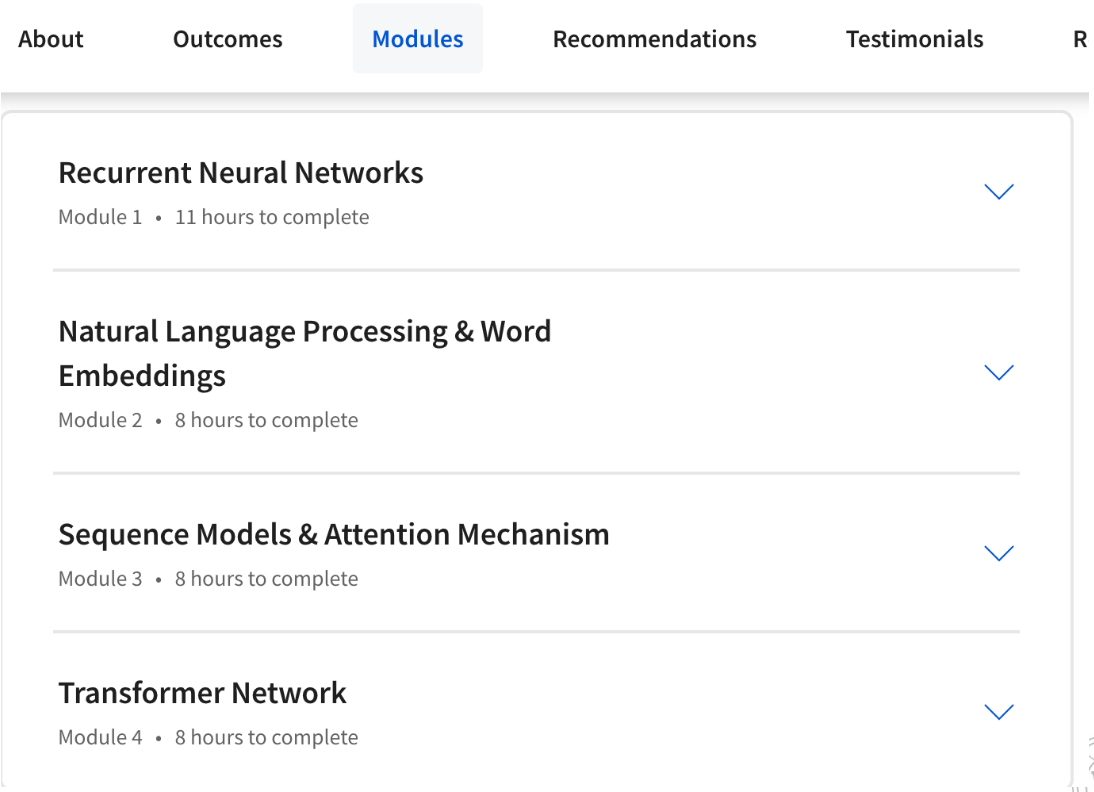
eBook pdf, ePub, online	**print** includes eBook	**online + audio** read and listen	**subscription** from $19.99

https://www.manning.com/books/deep-learning-with-python-second-edition

Additional Resources

Learn more about embeddings, sequence models, transformers, RNN, LSTM…

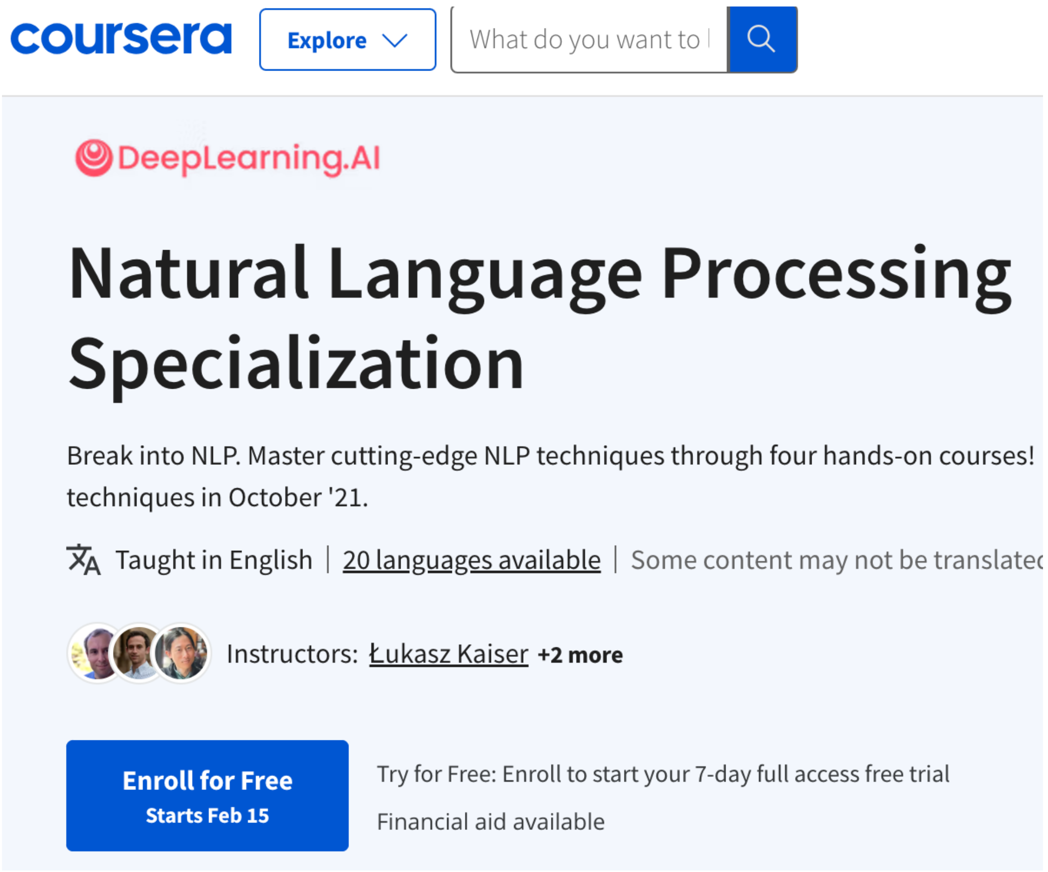


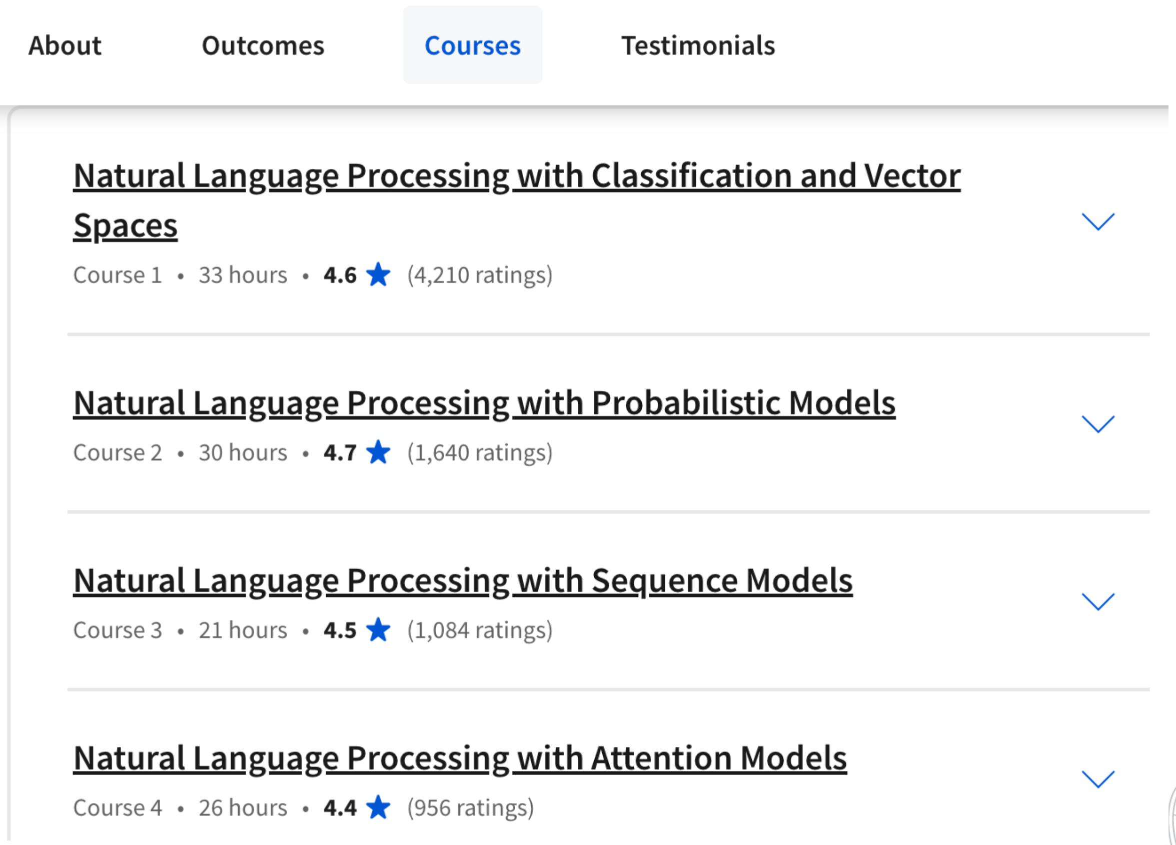


https://www.coursera.org/learn/nlp-sequence-models?specialization=deep-learning#modules

Additional Resources

Learn more about embeddings, sequence models, transformers, RNN, LSTM…





https://www.coursera.org/specializations/natural-language-processing#courses

Exercises using Google Colab