Entity Detection in the MIT Movie Corpus

Robert Mangan

July 1, 2020

Introduction

- The MIT Movie Corpus is a semantically tagged training and test corpus in BIO format.

Introduction

- The MIT Movie Corpus is a semantically tagged training and test corpus in BIO format.
- The BIO format is a format for 'chunks'.

Introduction

- The MIT Movie Corpus is a semantically tagged training and test corpus in BIO format.
- The BIO format is a format for 'chunks'.
- B: beginning of a chunk, I: inside a chunk, O: outside a chunk.

Introduction

- The MIT Movie Corpus is a semantically tagged training and test corpus in BIO format.
- The BIO format is a format for 'chunks'.
- B: beginning of a chunk, I: inside a chunk, O: outside a chunk.
- Each chunk refers to an entity such as 'plot', 'actor', etc.

```
O   show
O   me
O   films
O   with
B-ACTOR drew
I-ACTOR barrymore
O   from
O   the
B-YEAR  1980s
```

Introduction

- Dataset: eng corpus (simple queries), trivia10k13 corpus (210,000 complex queries)

Introduction

- Dataset: eng corpus (simple queries), trivia10k13 corpus (210,000 complex queries)
- Task: build a model to predict entities in the BIO format.

Introduction

- Dataset: eng corpus (simple queries), trivia10k13 corpus (210,000 complex queries)
- Task: build a model to predict entities in the BIO format.
- Approach: data preparation, feature engineering, model building and evaluation.

Introduction

- Dataset: eng corpus (simple queries), trivia10k13 corpus (210,000 complex queries)
- Task: build a model to predict entities in the BIO format.
- Approach: data preparation, feature engineering, model building and evaluation.
- Tools: Python, Pandas, NLTK, scikit-learn.

Preparing the data and feature extraction

- Add $<$START$>$ and $<$END$>$ tags at beginning and end of sentences, respectively.

Preparing the data and feature extraction

- Add <START> and <END> tags at beginning and end of sentences, respectively.
- Create features for previous and following words.

Preparing the data and feature extraction

- Add <START> and <END> tags at beginning and end of sentences, respectively.
- Create features for previous and following words.

	label	word	prevword	postword	feature_set
0	<START>	<START>	<END>	steve	{'word': '<START>', 'prevword': '<END>', 'post...
1	B-Actor	steve	<START>	mcqueen	{'word': 'steve', 'prevword': '<START>', 'post...
2	I-Actor	mcqueen	steve	provided	{'word': 'mcqueen', 'prevword': 'steve', 'post...
3	O	provided	mcqueen	a	{'word': 'provided', 'prevword': 'mcqueen', 'p...
4	O	a	provided	thrilling	{'word': 'a', 'prevword': 'provided', 'postwor...

Model I: Naive Bayes

- Train a Naive Bayes classifier.

Model I: Naive Bayes

- Train a Naive Bayes classifier.
- Accuracy on test set: 72% (current word), 78.5% (current and previous words), 80.5% (current, previous and following words)

Confusion Matrix

	I-Plot	O	<END>	<START>	B-Plot	I-Actor	B-Actor	I-Origin	B-Genre	B-Year
I-Plot	<29.9%>	2.7%	0.0%	0.0%	0.7%	0.1%	0.1%	0.2%	0.1%	0.1%
O	4.3%	<27.2%>	0.0%	0.0%	0.6%	0.1%	0.1%	0.2%	0.2%	0.1%
<END>	.	.	<4.5%>	.	.	.	.	.	.	.
<START>	.	.	.	<4.5%>	.	.	.	.	.	.
B-Plot	1.3%	1.0%	0.0%	0.0%	<1.3%>	0.0%	0.0%	0.0%	0.0%	0.0%
I-Actor	0.0%	0.1%	0.0%	0.0%	0.0%	<3.3%>	0.2%	0.0%	.	0.0%
B-Actor	0.0%	0.0%	0.0%	.	0.0%	0.0%	<2.9%>	0.0%	.	0.0%
I-Origin	0.3%	0.5%	.	0.0%	0.0%	0.1%	0.0%	<0.8%>	0.0%	0.0%
B-Genre	0.1%	0.1%	.	.	0.0%	0.0%	.	0.0%	<1.5%>	0.0%
B-Year	0.0%	0.0%	.	.	0.0%	.	.	0.0%	0.0%	<1.5%>

Model II: Naive Bayes (sequence)

- Create new feature for "previous label" and train a Naive Bayes Classifier.

Model II: Naive Bayes (sequence)

- Create new feature for "previous label" and train a Naive Bayes Classifier.
- For making predictions, use sequence approach. Assume we don't have access to the actual labels.

Model II: Naive Bayes (sequence)

- Create new feature for "previous label" and train a Naive Bayes Classifier.
- For making predictions, use sequence approach. Assume we don't have access to the actual labels.
- Classify the first instance using our model. Then use that predicted label as a feature for the following instance. Then make a prediction for that instance, and repeat.

Model II: Naive Bayes (sequence)

- Create new feature for "previous label" and train a Naive Bayes Classifier.
- For making predictions, use sequence approach. Assume we don't have access to the actual labels.
- Classify the first instance using our model. Then use that predicted label as a feature for the following instance. Then make a prediction for that instance, and repeat.
- Accuracy on test set: 81.2%.

Model III: Random Forest (word embeddings)

- Train a word embedding algorithm (based on movie corpus) using Word2Vec.

Model III: Random Forest (word embeddings)

- Train a word embedding algorithm (based on movie corpus) using Word2Vec.
- Convert each word to a vector and use these as input features. Also, use previous and following vectors as features.

Model III: Random Forest (word embeddings)

- Train a word embedding algorithm (based on movie corpus) using Word2Vec.
- Convert each word to a vector and use these as input features. Also, use previous and following vectors as features.
- Train a Random Forest Classifier.

Model III: Random Forest (word embeddings)

- Train a word embedding algorithm (based on movie corpus) using Word2Vec.
- Convert each word to a vector and use these as input features. Also, use previous and following vectors as features.
- Train a Random Forest Classifier.
- Accuracy on test set: 80%.

Potential improvements

- Recurrent Neural Network, LSTM

Potential improvements

- Recurrent Neural Network, LSTM
- Combine both datasets to have a larger training set.

Questions

Questions?