NER challenge

Author: Natalya Segal (natalya.segal@gmail.com)

Executive summary

I have trained Spacy NER model on Conll2003 English dataset as a baseline solution, resulting in F-score of 82.4%, with the best F-score in academic papers on this dataset being 93.5%.

Spacy was chosen for being fast to ramp up and highly efficient in production (more info in Spacy section of this doc). If we want to improve performance except for gathering our own data, I would try LSTM and BERT based models (see discussion below).

(Training notebook: *ner_proj/train_ner_spacy_conll2003.ipynb*, docker for inference *ner_proj/docker_ns*, instructions *ner_proj/Readme*, data parsing *ner_proj/notebooks/in parse_data.ipynb* notebook produces .csv files (in ner_proj/data). Those .csv files are used for exploratory data analysis in *ner_proj/notebooks/explore_data.ipynb* notebook.)

Repository: https://github.com/natalyasegal/spacy_ner_conll2003_en

Introduction

The goal of this challenge is to rapidly research, build, and deploy a machine learning system to extract entities from a piece of arbitrary text.

Our hypothetical business is a data broker that collects and analyzes travel data. We do this by scraping Twitter, travel forums, and other web sites to collect information on where people are traveling to and how many people are traveling together. We unfortunately don't have any annotated data of our own, so we'll be bootstrapping a model using publicly available data.

Given a piece of text, perform entity recognition on it to identify and extract entities. Possible entities could be things like: Person, Organization, Location, and Misc Names.

NER task

Named entity recognition (NER) is a well studied task, with several leading solutions taking a supervised approach and ongoing studies on NER as an unsupervised or semi-supervised task.

There are several pre-trained solutions for NER, including StanfordNER, Spacy and NLTK. Each taking a different approach and each pre-trained on a different dataset. Most of them allow to retrain the model on a dataset of interest. Here is a paper from Sept 2019 comparing their performance on different dataset including English part of CONLL2003.

There are also state of the art NER solutions, that are more computationally heavy and less production ready, but give a better F score on different datasets, including CONLL2003.

Leading academic solutions

CONLL2003 is a known dataset, so the leading NER solutions for this set are known (see Appendix A). Although we won't choose one of them for the baseline solution, it is still helpful to take a brief look at them.

The best performance (F1 score) is 93.5% for English dataset. Most of the leading academic solutions are either relatively complicated to implement, affecting time to market or will take a lot of computation resources in training and inference (like BERT), but the information above gives us a point of comparison as well as directions for further improvement that we'll discuss later in this document.

Proposed baseline solution based on Spacy

Having a limited time for the first production ready version, let's start with considering Spacy. Main advantages of Spacy for our task are: [1] it is fast to ramp up, [2] can be used with multiple languages, [3] production ready and time and space efficient in production, [4] very popular, so good quality documentation is available, [5] it runs on popular production platforms.

It's main disadvantage for the task is a low F1 score when using pre-trained version, that forces us to re-train the internal models on CONLL2003 dataset, and even when re-trained, we receive a slightly lower F1 score than in the leading academic solutions.

Here we have a tradeoff between F1 score and time/space complexity in production as well as time to market.

Spacy

Spacy is an open-source python library for NLP written in Python and Cython. It offers pre-trained models for multi-language NER, as well as allowing developers to train and deploy custom NER models on domain specific corpuses. SpaCy's pretrained models are trained on the OntoNotes 5 corpus. I'll train a new model on CONLL2003 dataset

If we use a pre-trained Spacy model, it performs relatively poorly (F score 65.14%) on the dataset of interest (see table1), but when trained on CONLL2003 English dataset gives F score of about 80% on CONLL2003 dataset test set (source), that is relatively good performance given that 93.5% is the known maximum on this set as for 2019.

Note: it may be worth diving deeper into the reasons for the poor performance of the pretrained model on another dataset that was detected in the reference above. It may either identify poor generalization or be a result of naive evaluation (OntoNotes 5 corpus format differs from one in CONLL2003). I would probably take a look at this issue in a real project before deciding on Spacy.

Data

The dataset is publicly available from Conll2003. You can learn more about it from here. Data cleaning and parsing is performed by ner_proj/notebooks/in parse_data.ipynb notebook, it produces .csv files (provided in ner_proj/data). Those .csv files are used for exploratory data analysis that can be viewed in ner_proj/notebooks/explore_data.ipynb notebook.

Conclusions from the brief data exploration are: [1] dataset is properly shuffled and entities types distribution in validation and test sets resemble those in the training set, [2] reasonable percent of data is left for validation set and test set, [3] no major tokenization problems, minor problem I have seen: splitting triling "'s", [4] haven't checked percent of mislabeling in annotation data, in real project would have re-annotated a small random sample of sentences to understand this present. [5]Good amount of examples for all labels, [6]
Major concern regarding using this dataset in our project is its origin, that implies a significant difference in percent of first person sentences related to our potential traffic that will originate from social media.

Training and validation data checkup is performed in the training notebook, the important point is that there is no overlap between training and validation set. Same about the test set (checked separately). The results:

```
=========================== Data format validation
===========================
✔ Corpus is loadable

=============================== Training stats
===============================
Training pipeline: tagger, parser, ner
Starting with base model 'en'
1499 training docs
347 evaluation docs
✔ No overlap between training and evaluation data

============================== Vocab & Vectors
==============================
ℹ 204567 total words in the data (23624 unique)
10 most common words: '.' (7374), ',' (7290), 'the' (7243), 'of' (3751),
'in'
(3398), 'to' (3382), 'a' (2994), '(' (2861), ')' (2861), 'and' (2838)
ℹ No word vectors present in the model
```

```
=========================== Named Entity Recognition
===========================
ℹ 2 new labels, 2 existing labels
0 missing values (tokens with '-' label)
New: 'LOC' (7140), 'PER' (6600), 'ORG' (6321), 'MISC' (3438)
Existing: 'ORG', 'LOC'
⚠ 15 entity span(s) with punctuation
✔ Good amount of examples for all labels
✔ Examples without occurrences available for all labels
✔ No entities consisting of or starting/ending with whitespace
Entity spans consisting of or starting/ending with punctuation can not be
trained with a noise level > 0.

============================ Part-of-speech Tagging
============================
ℹ 46 labels in data (50 labels in tag map)
'NNP' (34392), 'NN'
```

Training

Our baseline model (based on Spacy NER) is trained in
ner_proj/train_ner_spacy_conll2003.ipynb
(https://github.com/natalyasegal/spacy_ner_conll2003_en/blob/master/ner_proj/notebooks/train_ner_spacy_conll2003.ipynb)
This model can be trained on Conll2003 on a reasonably powerful CPU in reasonable time.

I have trained Spacy via CLI. Resulting Precision, Recall, F score:

```
NER Precision     82.16
NER Recall        82.79
NER F score       82.48
```

See the notebook above for more information.

The same notebook contains data conversion and practical exploration before training. See results summary in section above.

Note: Alternative training code of Spacy NER model can be borrowed from.
https://github.com/Djia09/Named-Entity-Recognition-spaCy. I haven't tried it.

Docker container for inference

The docker container can be found in ner_proj/docker_ns directory (https://github.com/natalyasegal/spacy_ner_conll2003_en/tree/master/ner_proj/docker_ns), instructions in Readme (https://github.com/natalyasegal/spacy_ner_conll2003_en/blob/master/README.md).

Rpc is based on flask. Spacy version matches one in training.
I have performed only basic optimizations on containers and additional changes / optimization are needed before production.

Before production we need:

- To define authorization
- Provide Jason based interface
- Consider a batch interface if needed
- Consider optimizations for container building time and container image size
- Consider wsgi

Generalization

In order to test whether the proposed solution will generalize well on the data from our traffic, we need to get a sample of the actual traffic, annotate it and test our solution on this sample.

If there is no way to collect the traffic sample, it may be worth testing performance on another publicly available dataset, one that we have not trained on. I would choose a dataset that was collected from websites and social media, since our hypothetical business is a data broker that collects and analyzes travel data from Twitter, travel forums, and other web sites.

Once we gather our own annotated data

Q. Once we gather our own annotated data, could we reuse this model or would we create a new one?

A. We can retrain our Spacy model using the same notebook with our own annotated dataset. Depending on the amount of annotated data we'll have in our own dataset, we'll decide whether to train only on the new dataset or combine it with our previous dataset. (or another NER dataset available).

Approaches to try after baseline solution

After the baseline model is ready, we can improve performance by adding more training data (adding data from available NER datasets), improving the quality of the training data (collecting our own dataset) and improving our model.

I would perform analysis of how well our model generalizes when trained on conll2003 and inference is performed on our traffic. It will be one factor affecting the decision whether to add/improve data or improve the model. Other factors will be availability of datasets to annotate, budget for annotation, budget for production vs requirements for quality.

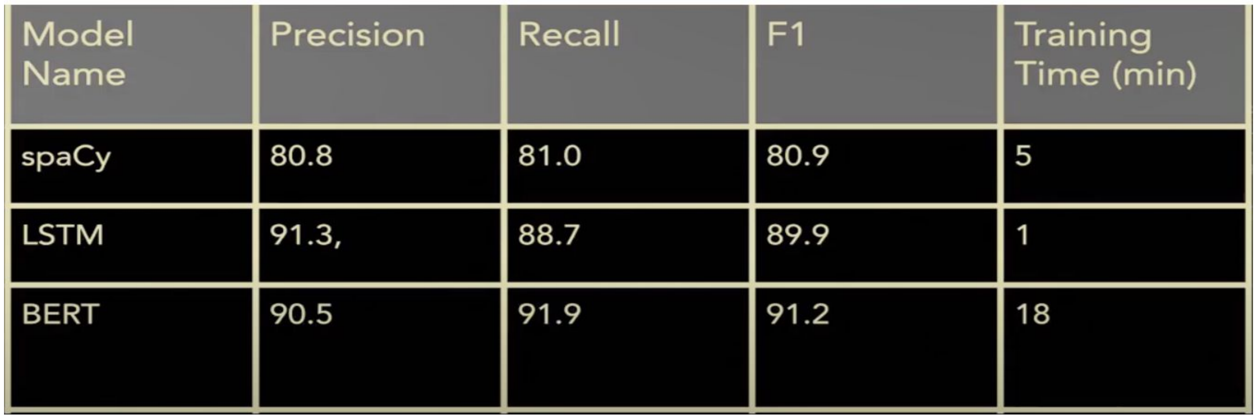
If we see poor generalization and can perform annotation in terms of budget and data availability, I'd opt for gathering our own annotated data. If we see reasonable generalization and/or have no way to gather our own annotated data, I'll proceed to an LSTM based model as it may give significant improvement in quality with relatively low penalty in production resources. (See discussion below) .

Disclaimer: I haven't trained LSTM on Conll2003, so everything written in this section is based on experience with LSTM and materials I have found on models performance on our dataset.

Discussion on Models

Let’s look at the approaches to explore after our baseline solution is in production. Papers from Appendix A suggest approaches worth trying are LSTM networks and variations of BERT.

A team that participated in ai.science workshop has tried those approaches during their workshop (see: video summary) with the following results for English Conll2003 dataset:



Model Name	Precision	Recall	F1	Training Time (min)
spaCy	80.8	81.0	80.9	5
LSTM	91.3,	88.7	89.9	1
BERT	90.5	91.9	91.2	18

We may get slightly different numbers (because of running on different versions and experimenting with parameters), so I bring the numbers above just to support a conclusion to try those directions. With Spacy, I have gotten slightly better results, than those in the table. Training time in this table is interesting only relative to each other.

While BERT has the best results of our dataset, BERT models are large, and typically warrant GPU acceleration. Working with GPUs can be expensive, and BERT will be slower to run on text than LSTM and much slower than tools like Spacy. So we need to consider production requirements for speed, accuracy, and cost before going straight to BERT. We also need to consider whether our inference will be performed online during time sensitive interaction with humans, or it may be performed online once in a period of lime (for instance once a day).

Training time should be also considered it we implement continuous training.

Comparison between LSTM and BERT

Both LSTM and BERT have good F scores on this set. Both have pros and cons. The main disadvantage of BERT is that it is heavy on resources in production even for inference. There are lighter versions of BERT that may be worth evaluating. The main advantages of BERT are quality and potential for generalization on other sets.

Main advantages of LSTM is a good quality with being relatively modest in resources requirements. It is heavier in production than Spacy, but significantly lighter than BERT. Main concern is ability to generalize to the actual traffic data.

So I would train an LSTM model on Conll2003, evaluate the generalization issue and then decide on the next step. If we see generalization problems, I would train LSTM on several datasets (ConLL2003 and several sets with data from social networks) and a set of annotated actual traffic if we can get one. If the set of actual traffic we can get annotated is too small, I would try transfer learning.

If after this effort, the model still generalizes poorly on our traffic and we cannot get a larger set of the actual traffic annotated, I would proceed to variations of BERT.

In case of unlimited production resources (typically not the case)

After the baseline solution is ready, I would go straight to BERT variations if we would like the best performance and are not limited in production resources. If extra improvement of performance is need, we can use combination of different models (see leading solutions in Appendix A)

Expanding Solution To Different languages

In the baseline model as well as in the best alternatives, there are no language specific feature engineering. It makes them good candidates to try when expanding to other languages. We need datasets in those languages and will train the suggested models on those new datasets. After training we'll get a trained model per language.

We can see that in some languages there may be the same winning approaches as in English, for example, as far as we can see from papers in Appendix A, B, the quality (precision, recall, f-score numbers) differ significantly between English and German, but there is a similarity between winning approaches.

If we expand for different languages, we'll need to evaluate the quality (precision, recall, F score) that we can get while extrapolating our leading approaches used previously for English (same libraries, model architectures, etc.) to those languages versus the best possible NER quality in those languages and see whether we can tolerate the penalty in quality at least in our baseline solution. Main advantages of following the same approach will be time to market and reuse of some of the code, cons - potentially non optimal quality.

If we can tolerate the penalty in quality, I would go for the same libraries and architectures for all the languages for our baseline solution and then explore other options if needed.

Appendix A:

There was a competition of leading academic NER approaches, evaluated on CONLL2003 English dataset. Results as well as papers presenting the solutions, can be found here.

Appendix B

NER approaches best performing on CONLL2003 in German from two sources:

1. Source 1:

Type	Model	Pr	R	F1
CRF	StanfordNER	74.18	72.50	73.33
RNN	Lample et al. (2016)	-	-	78.76
CRF	GermaNER	85.88	73.78	79.37
RNN	BiLSTM-WikiEmb	87.67	78.79	82.99*
RNN	BiLSTM-EuroEmb	79.92	72.14	75.83

   Source: https://www.aclweb.org/anthology/P18-2020.pdf, Table 3: Evaluation on the test set of

2. Source 2 - Multilingual Approaches

Classifier	Training set	Testset	F1-Score
LSTM-CRF (Lample 2016)	CoNLL-2003	CoNLL-2003	78.76
Multilingual Data for NER (Faruqui 2014)	CoNLL-2003 + Dutch CoNLL 2002	CoNLL-2003 Test?	77.37
Shallow Semi-Supervised Features (Agerri 2017)	CoNLL-2003	CoNLL-2003 Dev	77.18
Semi-Supervised Features (Agerri 2017)	CoNLL-2003	CoNLL-2003 Test	76.42

Source: https://github.com/MaviccPRP/ger_ner_evals

Appendix C - Annotations

There are different ways to make the annotations, some organizations employ in-house annotators, those that do not - use external services. Example of such a service is Figure Eight.

Tips for working with annotation services:

1. When planning a budget for an annotation task we should pay attention that we'll make a majority vote between several annotators, for example 2 out of 3 annotators should agree for us to accept the annotation. So the task will be done at least 3 times (by 3 different people), therefore we need to factor it into the budget.
2. We need to add test questions, such as annotation of a sentence that we already have annotated. The purpose is to test the annotators on them. This is required as we do not know anything about people that perform the job and would like to catch problems as early as possible.

3. It is important to define the annotation task as simple as possible