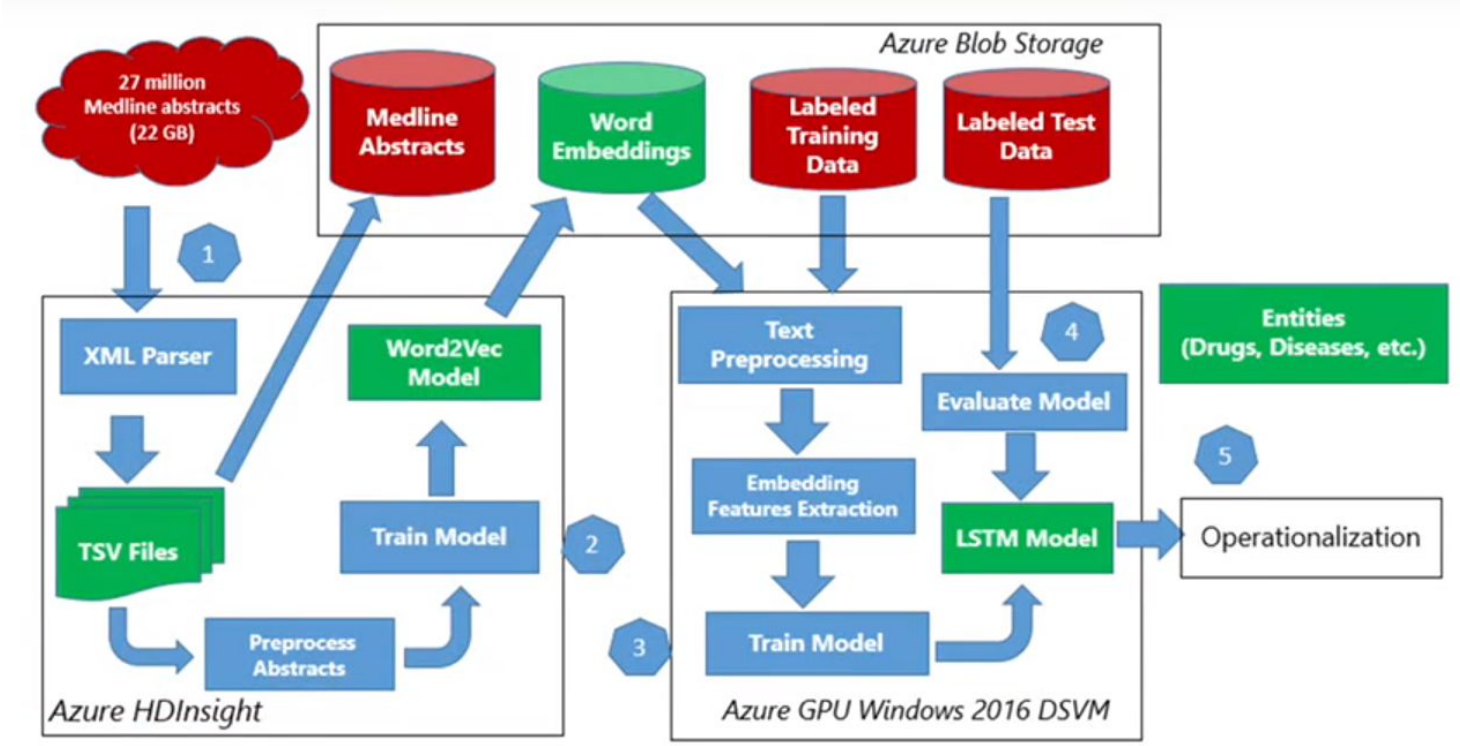
Entity Extraction is a subtask of a bigger domain called Information Extraction.
Also called as Named-Entity Recognition (NER), entity chunking and entity identification.

Find out the entity and the confidence.

Approach

1. Feature Extraction Phase - Domain Specific Features
   Use a large amount of unlabeled domain-specific data corpus such as Medline PubMed abstracts to train a neural (LSTM may be) word embedding model.
2. Model Training Phase - Domain Specific Model
   The output embeddings are considered as automatically generated features to train a neural entity extractor using a small/reasonable amount of labeled data.



Step 1: Word Embedding

Input: Words

If you want to feed texts into a deep learning model. At first it has to be converted into vector representation of numbers of some dimension.

Let's suppose there are word embeddings of 100s dimension and we want to find the closest words to the word “Brain”.

These algorithms tries to assign similar vector to the words in the same context.
The training data for these words are the context surrounding these words. So that is why it is an unsupervised algorithm.
So we need sentences and in sentences we find text which makes the contexts for these words. So, it's not just syntactic similarity its a semantic similarity.

Custom Word Embeddings:
Publically available pre trained models such as Google News. But can it be used on a domain specific entity extraction.
Train a model on data
Word2Vec algorithm
Now the word embedding model is ready

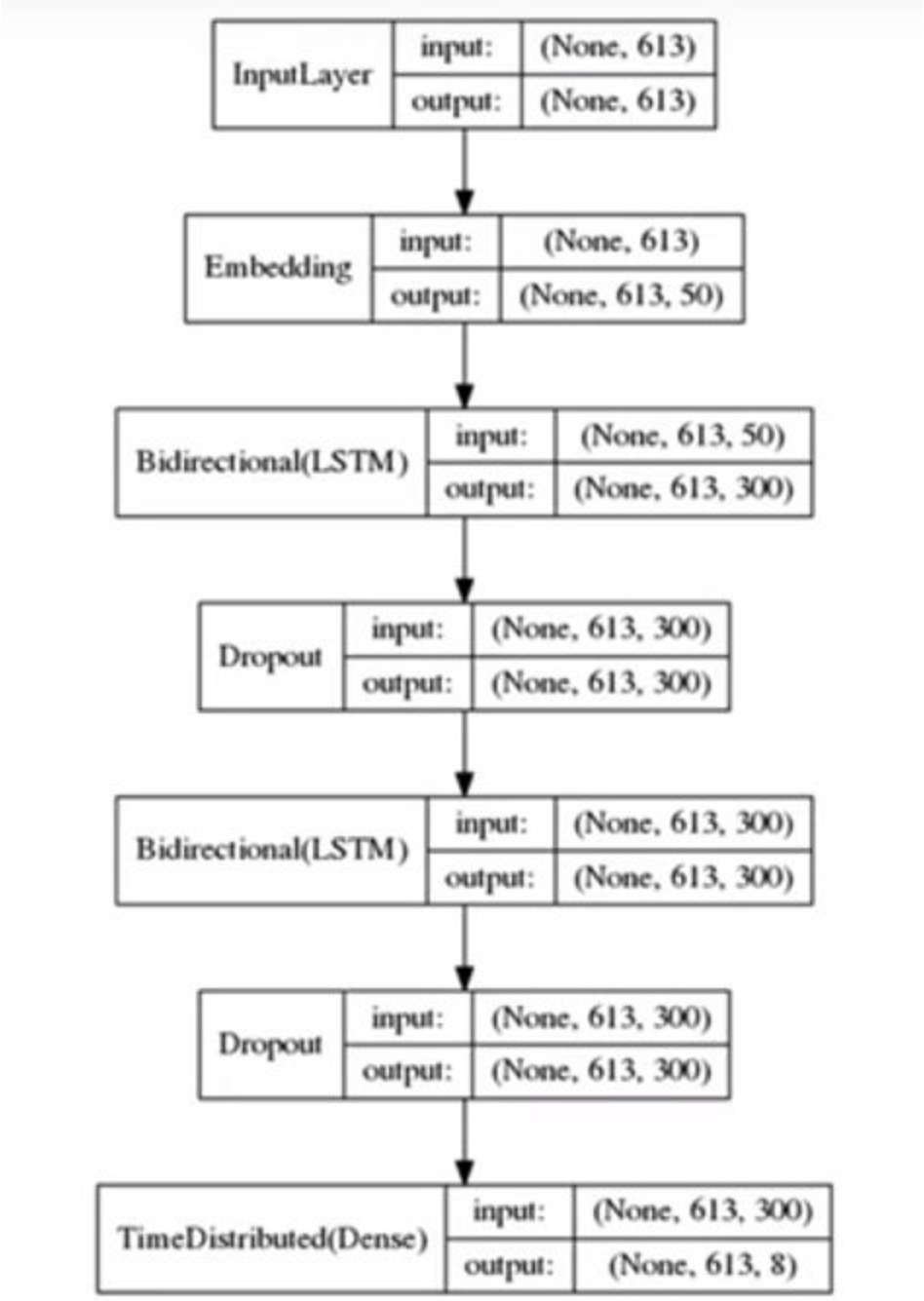
Step 2: Use this Word embedding as features to train an entity extraction model

Word embedding model converts texts (training data) into vectors which is used as feature for Entity extraction model.

Why Deep Learning

- Can find complex relationships between input and output using
  - Non-Linear processing units
  - Multiple Hidden layers

- Special-purpose layers/architectures have been developed for different problems
  - RNN are commonly used for sequence-labeling tasks
  - Long Short-Term Memory layer (LSTM)
- Comparison to Conditional Random Fields (CRFs)
  - CRFs are linear models w/o hidden layers
  - CRFs have short-term memory



Deep Neural Network Architecture

Embedding Layer: Act as an embedding for the input text.

Two LSTM Layer: Which improves performance over one single layer

Two Dropout layers: It basically prunes the model, so it doesn't overfits the data

Long short Term Memory (LSTM). But not any type of LSTM, we need to use bi-directional LSTMs because using a standard LSTM to make predictions will only take the "past" information in a sequence of the text into account. for NER, since the context covers past and future labels in a sequence, we need to take both the past and the future information into account. **A bidirectional LSTM is a combination of two LSTMs — one runs forward from "right to left" and one runs backward from "left to right".**

Dataset Description

- Chemicals and Diseases Detection
- BioCreative V CDR task corpus, 2015
  http://www.biocreative.org/tasks/biocreative-v/track-3-cdr/
- Training Data Stats
  - # words = 214,198
  - # disease entities = 7699
  - # chemical entities = 9575
- Test Data Stats
  - # words = 112,285
  - # disease entities = 4080
  - # chemical entities = 4941

Other algorithm to compare with:

Conditional Random Fields (CRF)

CRFSuite:

- Extract traditional features
- Train CRF model

Traditional Features	
Word	lowercased token
Orthographic	upperInitial, allCaps, lowercase, mixedCaps
TokenType	word, number, symbol, punctuation
POS	part-of-speech tag

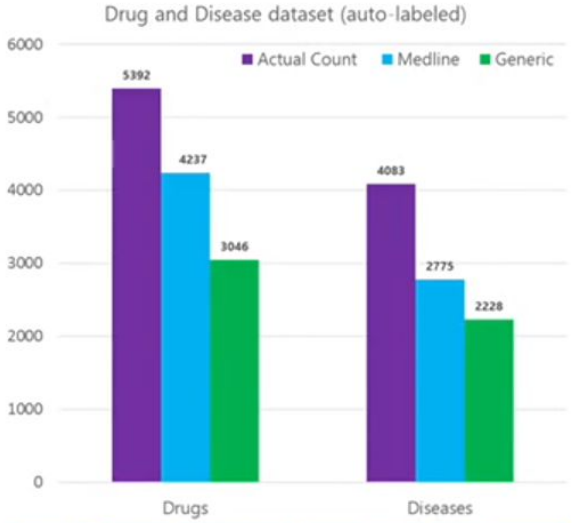
This models are good but advantage with the above model Word Embedding + LSTM is that you don't have to worry about the Feature Engineering.

Results (exact match)

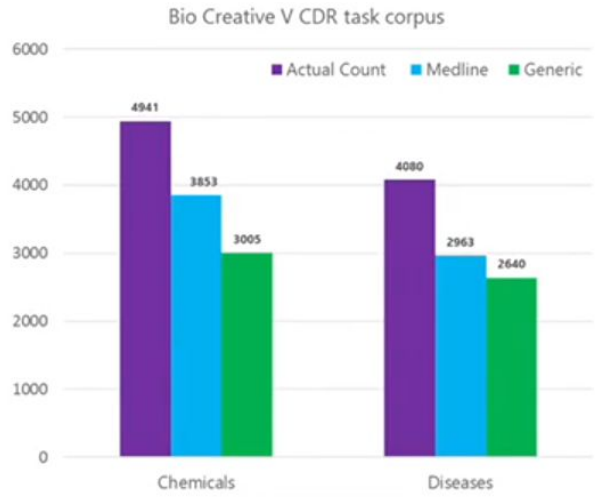
Algorithm + Features	Recall	Precision	F-score
Dictionary Lookup	64%	74%	68%
CRF: Traditional Features	61%	**81%**	70%
CRF: Pubmed Embedding	40%	61%	48%
CRF: Traditional + Pubmed Embed.	65%	80%	71%
LSTM: Pubmed Embedding	**76%**	77%	**76%**
LSTM: Generic Embeddings	74%	63%	67%

Comparison of domain specific embedding with the generic google news embedding.

Embedding Comparison



Embedding Dataset	Precision	Recall	F1 Score
MEDLINE	**0.73**	**0.74**	**0.74**
Google News	0.68	0.55	0.61



Embedding Dataset	Precision	Recall	F1 Score
MEDLINE	**0.77**	**0.76**	**0.76**
Google News	0.74	0.63	0.67

Takeaways

- Recipe for building a custom entity extraction pipeline:
  - Get a large amount of in-domain unlabeled data
  - Train a word2vec model on unlabeled data on Spark
  - Get as much of labeled data as possible
  - Train an LSTM -based Neural Network on a GPU-enabled machine
- Word embeddings are powerful features
  - Convey word semantics
  - Perform better than traditional features
  - No feature engineering
- LSTM NN is more powerful model than traditional CRF

Conditional Random Fields

-CRF for short
-Discriminative HMM with fewer independence assumptions
-How does it work though?

HMM - Hidden Markov Model