Learning to Extract

CSCI-GA.2590

Ralph Grishman
Flavors of learning

• Supervised learning
  – All training data is labeled
• Semi-supervised learning
  – Part of training data is labeled (‘the seed’)
  – Make use of redundancies to learn labels of additional data, then train model
  – Co-training
  – Reduces amount of data which must be hand-labeled to achieve a given level of performance
• Active learning
  – Start with partially labeled data
  – System selects additional ‘informative’ examples for user to label
Semi-supervised learning

L = labeled data
U = unlabeled data

1. L = seed
   -- repeat 2-4 until stopping condition is reached
2. C = classifier trained on L
3. Apply C to U.
   N = most confidently labeled items
4. L += N; U -= N
**Confidence**

How to estimate confidence?

- **Binary probabilistic classifier**
  - Confidence = | P – 0.5 | * 2

- **N-ary probabilistic classifier**
  - Confidence = P₁ – P₂

  where
  - P₁ = probability of most probable label
  - P₂ = probability of second most probable label

- **SVM**
  - Distance from separating hyperplane
Co-training

- Two ‘views’ of data (subsets of features)
  - Producing two classifiers $C_1(x)$ and $C_2(x)$
- Ideally
  - Independent
  - Each sufficient to classify data
- Apply classifiers in alternation (or in parallel)
  1. $L = \text{seed}$
     -- repeat 2-7 until stopping condition is reached
  2. $C_1 = \text{classifier trained on } L$
  3. Apply $C_1$ to $U$
     $N = \text{most confidently labeled items}$
  4. $L += N; \ U -= N$
  5. $C_2 = \text{classifier trained on } L$
  6. Apply $C_2$ to $U$
     $N = \text{most confidently labeled items}$
  7. $L += N; \ U -= N$
Problems with semi-supervised learning

• When to stop?
  • U is exhausted
  • Reach performance goal using held-out labeled sample
  • After fixed number of iterations based on similar tasks

• Poor confidence estimates
  • Errors from poorly-chosen data rapidly magnified
Learning Names

• We have discussed hand-coded rules and supervised models (HMM, MEMM, CRF) for NER [named entity recognition]

• We will now consider
  • Semi-supervised models
  • Active learning
Semi-supervised NER

• Annotating a large corpus to train a high-performance NER is fairly expensive

• We can use the idea of name consistency across documents to train an NER using
  – A smaller annotated corpus
  – A large unannotated corpus
Co-training for NER

• We can split the features for NER into two sets:
  – Spelling features
    (the entire name + tokens in the name)
  – Context features
    (left and right contexts + syntactic context)

• Start with a seed
  – E.g., some common unambiguous full names

• Iteratively grow seed, alternatively applying spelling and context models and adding most-confidently-labeled instances to seed
Co-training for NER

1. **Seed**
2. **Build context model**
3. **Apply context model**
4. **Add most confident exs to labeled set**
5. **Build spelling model**
6. **Apply spelling model**
7. **Add most confident exs to labeled set**
8. **Build context model**
9. **Apply context model**
Name co-training: results

- 3 classes: person, organization, location (and ‘other’)
- Data: 1M sentences of news
- Seed:
  - New York, California, U.S. → location
  - contains(Mr.) → person
  - Microsoft, IBM → organization
  - contains(Incorporated) → organization
- Took names appearing with appositive modifier or as complement of preposition (88K name instances)
- Accuracy: 83%
- Clean accuracy (ignoring names not in one of the 3 categories): 91%

(Collins and Singer 1999)
Semi-supervised NER: when to stop

• Semi-supervised NER labels a few more examples at every iteration
  – It stops when it runs out of examples to label

• This is fine if
  – Names are easily identified (e.g., by capitalization in English)
  – Most names fall into one of the categories being trained (e.g., people, organizations, and locations for news stories)
Semi-supervised NER: semantic drift

• Semi-supervised NER doesn’t work so well if
  – The set of names is hard to identify
    • Monocase languages
    • Extended name sets including lower-case terms
  – The categories being trained cover only a small portion of the set of names

• The result is *semantic drift* and *semantic spread*
  – The name categories gradually grow to include related terms
Fighting Semantic Drift

• We can fight drift by training a larger, more inclusive set of categories
  – Including ‘negative’ categories
    • Categories we don’t really care about but include to compete with the original categories
  – These negative categories can be built
    • By hand (Yangarber et al. 2003)
    • Or automatically (McIntosh 2010)
Active Learning

• For supervised learning, we typically annotate text data sequentially

• Not necessarily the most efficient approach
  • Most natural language phenomena have a Zipfian distribution ... a few very common constructs and lots of infrequent constructs
  • After you have annotated “Spain” 50 times as a location, the NER model is little improved by annotating it one more time

• We want to select the most informative examples and present them to the annotator
  • The data which, if labeled, is most likely to reduce NER error
How to select informative examples?

- Uncertainty-based sampling
  - For binary classifier
    - For MaxEnt, probability near 50%
    - For SVM, data near separating hyperplane
  - For n-ary classifier, data with small margin

- Committee-based sampling
  - Data on which committee members disagree
  - (co-testing ... use two classifiers based on independent views)
Representativeness

• It’s more helpful to annotate examples involving common features
  • Weighting these features correctly will have a larger impact on error rate

• So we rank examples by frequency of features in the entire corpus
Batching and Diversity

• Each iteration of active learning involves running classifier on (a large) unlabeled corpus
  – This can be quite slow
  – Meanwhile annotator is waiting for something to annotate

• So we run active learning in batches
  – Select best $n$ examples to annotate each time
  – But all items in a batch are selected using the same criteria and same system state, and so are likely to be similar

• To avoid example overlap, we impose a diversity requirement with a batch: limit maximum similarity of examples within a batch
  – Compute similarity based on example feature vectors
Simulated Active Learning

• True active learning experiments are
  – Hard to reproduce
  – Very time consuming

• So most experiments involve *simulated active learning*:
  – “unlabeled” data has really been labeled, but the labels have been hidden
  – When data is selected, labels are revealed
  – Disadvantage: “unlabeled” data can’t be so bit

• This leads us to ignore lots of issues of true active learning:
  – An annotation unit of one sentence or even one token may not be efficient for manual annotation
  – So reported speed-ups may be optimistic (typical reports reduce by half the amount of data to achieve a given NER accuracy)
Limitations

• Cited performance is for well matched training and test
  • Same domain
  • Same source
  • Same epoch
  – Performance deteriorates rapidly if less matched
    • NER trained on Reuters (F=91),
      tested on Wall Street Journal (F=64) [Ciaramita and Altun 2003]
  – Work on NER adaptation is vital

• Adding rarer classes to NER is difficult
  – Supervised learning inefficient
  – Semi-supervised learning is subject to semantic drift
Learning Relations

• How do these models apply to training relation extractors?
Supervised learning

• Collect training data
  – Annotate corpus with entities and relations
  – For every pair of entities in a sentence
    • If linked by a relation, treat as positive training instance
    • If not linked, treat as a negative training instance

• Train model
  – For $n$ relation types, either
    • Binary (identification) model + $n$-way classifier model or
    • Unified $n+1$-way classifier

• On test data
  – Apply entity classifier
  – Apply relation classifier to every pair of entities in same sentence
Types of Classifiers

• MaxEnt

• Kernel-based

• Deep learning
  – CNN [convolutional NN]
  – RNN [recurrent NN]
Supervised relation learner: features

- Heads of entities
- Types of entities
- Distance between entities
- Containment relations
- Word sequence between entities
- Individual words between entities
- Dependency path
- Individual words on dependency path
Kernel Methods

• Goal is to find training examples similar to test case
  – Similarity of word sequence or tree structure
  – Determining similarity through features is awkward
  – Better to define a similarity measure directly: a kernel function

• Kernels can be used directly by
  – SVMs
  – Memory-based learners (k-nearest-neighbor)

• Kernels defined over
  – Sequences
  – Parse or Dependency Trees
Putting it Together
Deep Learning Methods

• Don’t require specification of features
• Top performance (by a few %)
• CNN effective because most relations are local
Semi-supervised methods

• Preparing training data is more costly than for names
  – Must annotate entities and relations
• So there is a strong motivation to minimize training data through semi-supervised methods
• As for names, we will adopt a co-training approach:
  – Feature set 1: the two entities
  – Feature set 2: the contexts between the entities
• We will limit the bootstrapping
  – to a specific pair of entity types
  – and to instances where both entities are named
Semi-supervised learning

• Seed:
  • *[Moby Dick, Herman Melville]*

• Contexts for seed:
  • ... wrote ...
  • ... is the author of ...

• Other pairs appearing in these contexts
  • *[Animal Farm, George Orwell]*
  • *[Don Quixote, Miguel de Cervantes]*

• Additional contexts ...
Co-training for relations

- Find occurrences of seed tuples
- Tag entities
- Generate new seed tuples
- Generate extraction patterns
- Seed
Ranking contexts

• If relation R is functional, and [X, Y] is a seed, then [X, Y'], Y' ≠ Y, is a negative example

• Confidence of pattern P

\[ \text{Conf}(P) = \frac{P.\text{positive}}{P.\text{positive} + P.\text{negative}} \]

• where

\[ P.\text{positive} = \text{number of positive matches to pattern P} \]
\[ P.\text{negative} = \text{number of negative matches to pattern P} \]
Ranking pairs

• Once a confidence has been assigned to each pattern, we can assign a confidence to each new pair based on the patterns in which it appears
  – Confidence of best pattern
  – Combination assuming patterns are independent

\[
\text{Conf}(X,Y) = 1 - \prod_{P \in \text{contexts of } (X,Y)} (1 - \text{Conf}(P))
\]
Semantic drift

• Ranking / filtering quite effective for functional relations (book → author, company → headquarters)
  – But expansion may occur into other relations generally implied by seed (‘semantic drift’)
    • Ex: from governor → state governed to person → state born in

• Precision poor without functional property
Distant supervision

• Sometimes a large data base is available involving the type of relation to be extracted
  • A number of such public data bases are now available, such as FreeBase and Yago

• Text instances corresponding to some of the data base instances can be found in a large corpus or from the Web

• Together these can be used to train a relation classifier
Distant supervision: approach

• Given:
  • Data base for relation R
  • Corpus containing information about relation R
• Collect \( \langle X, Y \rangle \) pairs from data base relation R
• Collect sentences in corpus containing both X and Y
  • These are positive training examples
• Collect sentences in corpus containing X and some \( Y' \) with the same entity type as Y such that \( \langle X, Y' \rangle \) is not in the data base
  • These are negative training examples
• Use examples to train classifier which operates on pairs of entities
Distant supervision: limitations

• The training data produced through distant supervision may be quite noisy:
• If a pair <X, Y> is involved in multiple relations, R<X, Y> and R’<X, Y> and the data base represents relation R, the text instance may represent relation R’, yielding a false positive training instance
  – If many <X, Y> pairs are involved, the classifier may learn the wrong relation
• If a relation is incomplete in the data base ... for example, if resides_in<X, Y> contains only a few of the locations where a person has resided ... then we will generate many false negatives, possibly leading the classifier to learn no relation at all
Evaluation

• Matching relation has matching relation type and arguments
  – Count correct, missing, and spurious relations
  – Report precision, recall, and F measure

• Variations
  – Perfect mentions vs. system mentions
    • Performance much worse with system mentions
      – an error in either mention makes relation incorrect
  – Relation type vs. relation subtype
  – Name pairs vs. all mentions
    • Bootstrapped systems trained on name-name patterns

• Best ACE systems on perfect mentions: F = 65