Assigning Lexical Types to Unknown Words

João Silva

NLX—Natural Language and Speech Group
http://nlx.di.fc.ul.pt
University of Lisbon

Delph-In Meeting
July 2010

Presentation outline

Introduction

The problem

- LX-Gram has low coverage

Objective

- Increase the coverage of the grammar
- Make the grammar robust to OOV words

The approach

- Assign lexical types on-the-fly prior to parsing
  (ideally, a single type for each OOV word)
- Use machine learning methods

Datasets

Dataset creation

- Deep databank produced with LX-Gram
- Mostly newspaper texts, but also test suites (for regression)
- A set of tools for extracting *vistas*:
  TreeBank, DepBank, PropBank, etc.

Some numbers

- Version 2 of the databank
- $1,204$ sentences for a total of $9,789$ tokens
- 274 different lexical types
  - Highly skewed
    $\approx$ 50% occur at most 4 times
    $\approx$ 25% occur only once

TnT supertagger

Use a POS-tagger

- TnT: Second-order Markov Models
- Train and tag with default parameters
- Dataset: sentences, tokens are lemmas tagged with lexical types

Evaluation

- 10-fold cross-evaluation, 90–10% split
- Accuracy:
  - 88.58% (over all tokens)
  - 42.22% (over unknown tokens)

C&C supertagger

Use a supertagger

- C&C: Maximum-entropy model
- Train and tag with default parameters
- Dataset: sentences, tokens are lemmas tagged with POS and lexical types

Evaluation

- 10-fold cross-evaluation, 90–10% split
- Accuracy: 79.61% (over all tokens, gold POS tags)

Supertagger comparison

Summary

- TnT is a POS tagger with a simple trigram model
- C&C is a supertagger, uses POS tags, etc.
- However, TnT (88.58%) beats C&C (79.61%)

Supertagger comparison

Summary

- TnT is a POS tagger with a simple trigram model
- C&C is a supertagger, uses POS tags, etc.
- However, TnT (88.58%) beats C&C (79.61%)

Surprising?

Results from (Dridan, 2009):

- TnT (91.47%) beats C&C (89.08%)
- Learning curves:
  TnT has stabilized but C&C is still rising

TiMBL classifier

Dedicated classifier

- TiMBL: Memory-based learner
- Dataset: for each word, a set of 26 features including lemma, POS, previous POS, dependents, etc.
- 10-fold cross-evaluation, 90–10% split

A classifier for all types

- Accuracy: 79.37% (over all tokens)
- For VERB.DIR_TRANS: AUC: 0.7895

Set of binary classifiers

- One classifier for each lexical type
- For VERB.DIR_TRANS: AUC: 0.8041

Imbalanced datasets

Imbalanced datasets are a major problem for ML algorithms

- Issues in training and evaluation

Methods for dealing with imbalanced datasets:

- Under-sampling the majority class
- Over-sampling the minority class
- SMOTE (Synthetic Minority Over-sampling TEchnique)
  Creates new synthetic examples for the minority class

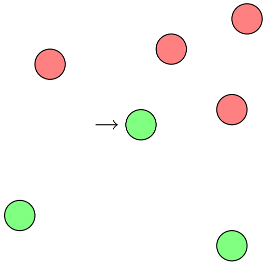
Imbalanced datasets

Imbalanced datasets are a major problem for ML algorithms

- Issues in training and evaluation

Methods for dealing with imbalanced datasets:

- Under-sampling the majority class
- Over-sampling the minority class
- SMOTE (Synthetic Minority Over-sampling TEchnique)
  Creates new synthetic examples for the minority class



1. Take an example from the minority class

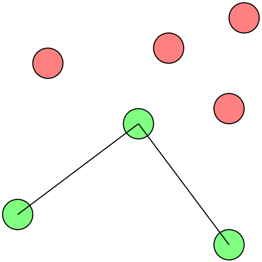
Imbalanced datasets

Imbalanced datasets are a major problem for ML algorithms

- Issues in training and evaluation

Methods for dealing with imbalanced datasets:

- Under-sampling the majority class
- Over-sampling the minority class
- SMOTE (Synthetic Minority Over-sampling TEchnique)
  Creates new synthetic examples for the minority class



1. Take an example from the minority class
2. Link to k nearest minority neighbors

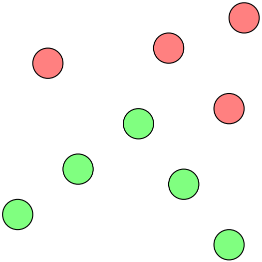
Imbalanced datasets

Imbalanced datasets are a major problem for ML algorithms

- Issues in training and evaluation

Methods for dealing with imbalanced datasets:

- Under-sampling the majority class
- Over-sampling the minority class
- SMOTE (Synthetic Minority Over-sampling TEchnique)
  Creates new synthetic examples for the minority class



1. Take an example from the minority class
2. Link to k nearest minority neighbors
3. Create a new case along each link

Applying SMOTE

For the (binary) VERB.DIR_TRANS dataset

- SMOTE implementation in Weka, default parameters
- Doubled the number of positive examples
  (108 in 9789 $\rightarrow$ 216 in 9897)

TiMBL: Before and after SMOTE

AUC: 0.8041 (worse than TnT)	$\longrightarrow$	AUC: 0.9148 (matches TnT)

Applying SMOTE

For the (binary) VERB.DIR_TRANS dataset

- SMOTE implementation in Weka, default parameters
- Doubled the number of positive examples
  (108 in 9789 $\rightarrow$ 216 in 9897)

TiMBL: Before and after SMOTE

AUC: 0.8041 (worse than TnT) $\longrightarrow$ AUC: 0.9148 (matches TnT)

A great improvement, but…

The newly created synthetic examples aren't linguistically sound!
(e.g. punctuation token with a verb POS feature)

Open questions and future work

The main question

Can a “supertagger” be better than the grammar?

Open questions and future work

The main question

Can a “supertagger” be better than the grammar?

Future work

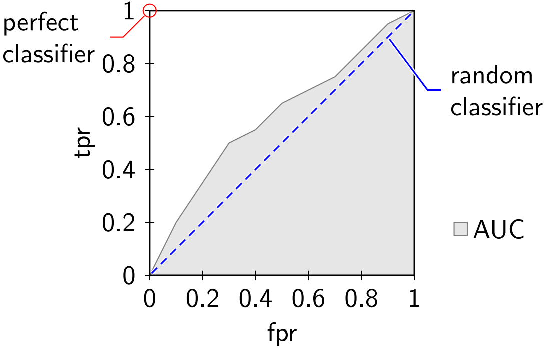
- Move on to the next stable version of the databank
- Test more tools, better features, etc.
- *n*-best supertaggers
- Linguistically-aware SMOTE
- Integrate into PET
- . . .

Thank you!

AUC: Area Under (ROC) Curve

ROC (Receiver Operating Characteristics) graph:
Shows tradeoff between hit rates (tpr) and false alarm rates (fpr)

$$\text{tpr} = \frac{\text{positives correctly classified}}{\text{total positives}} \qquad \text{fpr} = \frac{\text{negatives incorrectly classified}}{\text{total negatives}}$$



Return