GATE and Social Media: Normalisation and PoS-tagging

Leon Derczynski
Kalina Bontcheva

Tweet Normalisation

- “RT @Bthompson WRITEZ: @libbyabrego honored?! Everybody knows the libster is nice with it...lol...(thankkkks a bunch;))”
- OMG! I’m so guilty!!! Sprained biibii’s leg! ARGHHHHHH!!!!!!
- Similar to SMS normalisation
- For some components to work well (POS tagger, parser), it is necessary to produce a normalised version of each token
- BUT uppercasing, and letter and exclamation mark repetition often convey strong sentiment
- Therefore some choose not to normalise, while others keep both versions of the tokens

Lexical normalisation

- Two classes of word not in dictionary
  - 1. Mis-spelled dictionary words
  - 2. Correctly-spelled, unseen words (e.g. foreign surnames)
  - Problem: Mis-spelled unseen words (these can be in the dict!)
- First challenge: separate out-of-vocabulary and in-vocabulary
- Second challenge: fix mis-spelled IV words
  - Edit distance (e.g. Levenshtein): count character adds, removes

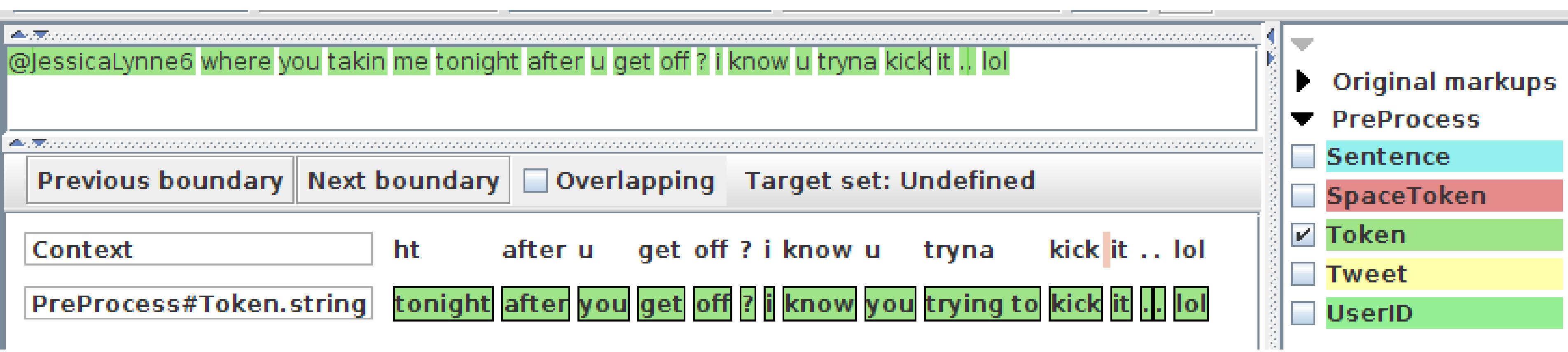
    zseged → szeged (distance = 2)
  - Pronunciation distance (e.g. double metaphone):

    YEEAAAHHH → yeah
- Need to set bounds on these, to avoid over-normalising OOV words

GATE Tweet Normaliser

- Create an instance of the Tweet Normaliser PR
  - If you don't have it available, load the Twitter plugin via plugin manager
- Add the Twitter Normaliser PR at the end of your pipeline
- Set its inputAS and outputAS params to **PreProcess**
- Create a new corpus and save to DS
- Populate it from a single file: **corpora/normaliser_test_corpus.xml**
  - Remember to specify root element **doc_root**
- Run the pipeline and inspect the results
  - Check the features on **Token** annotations

A normalised example



- Normaliser currently based on spelling correction and some lists of common abbreviations
- Outstanding issues:
  - Insert new Token annotations, so easier to POS tag, etc? For example: “trying to” now 1 annotation
  - Some abbreviations which span token boundaries (e.g. gr8, do n’t) are not yet handled
  - Capitalisation and punctuation normalisation

Part-of-speech tagging: example

Many unknowns:

- Music bands: Soulja Boy | TheDeAndreWay.com in stores Nov 2, 2010
- Places: #LB #news: Silverado Park Pool Swim Lessons

Capitalisation way off

- @thewantedmusic on my tv :) aka derek
- last day of sorting pope visit to birmingham stuff out
- Don't Have Time To Stop In??? Then, Check Out Our Quick Full Service Drive Thru Window :)

Part-of-speech tagging: example

Slang

- **~HAPPY <u>B-DAY</u> TAYLOR !!! <u>LUVZ</u> <u>YA</u>~**

Orthographic errors

- **dont even have <u>homwork</u> today, <u>suprising</u>?**

Dialect

- **Shall we go out for dinner this evening?**
- **fancy a cheeky nandoz tho**
  - ***Can I have a go on your iPad?***

Part-of-speech tagging: issues

Low performance

- Using in-domain training data, per token: SVMTool 77.8%, TnT 79.2%, Stanford 83.1%
- Whole-sentence performance: best was 10%; cf. SotA on newswire about 55-60%

Problems on unknown words – this is a good target set to get better performance on

- 1 in 5 words completely unseen
- 27% token accuracy on this group

Part-of-speech tagging: issues

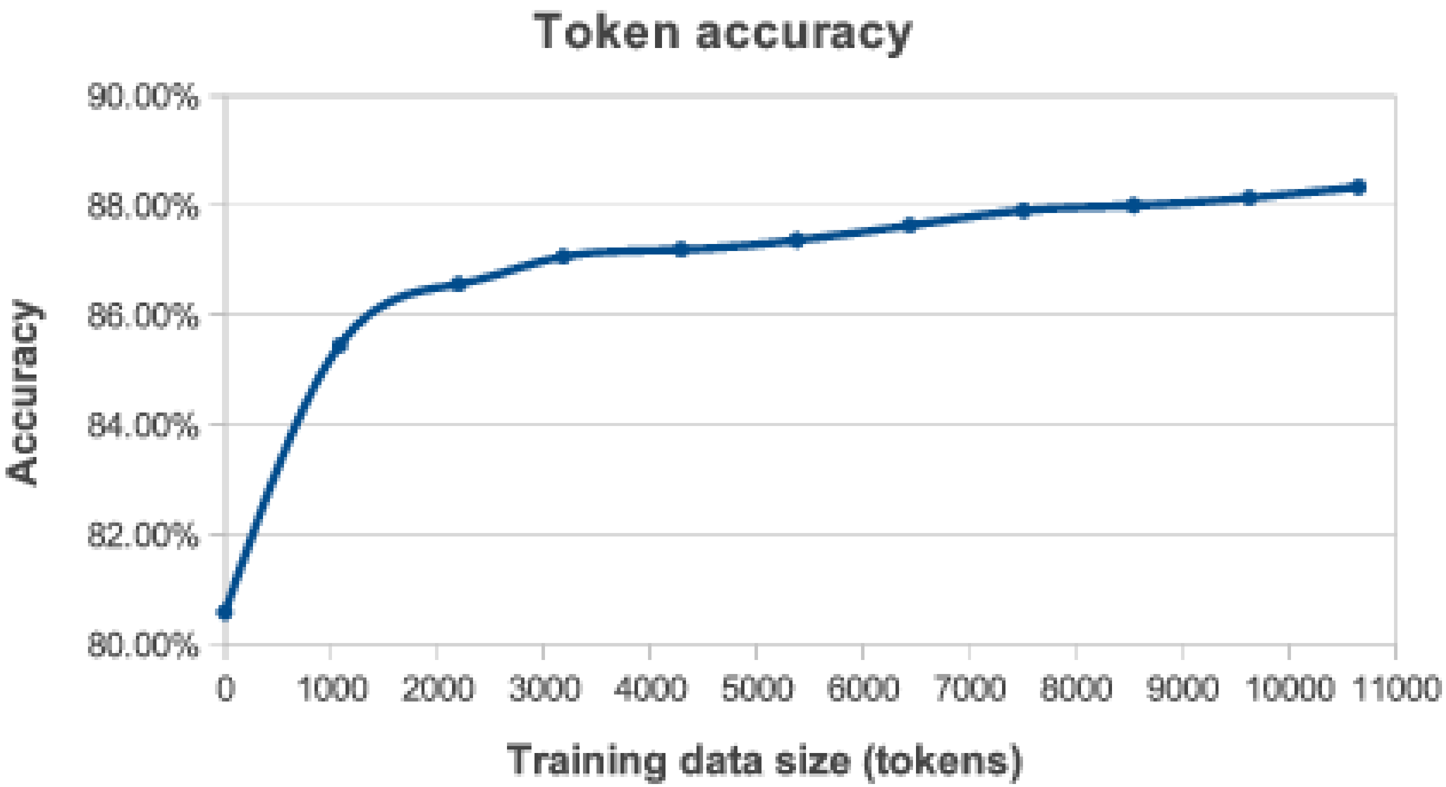
Errors on unknown words

- Gold standard errors (dank_UH je_UH → _FW) (Plank 2014)
- Training lacks IV words (Internet, bake)
- Pre-taggables (URLs, mentions, retweets)
- NN vs. NNP (derek_NN, Bed_NNP)
- Slang (LUVZ, HELLA, 2night)
- Genre-specific (unfollowing)
- Leftover tokenisation errors (ass**sneezes)
- Orthographic (suprising)

Part-of-speech tagging: issues

Insufficient data

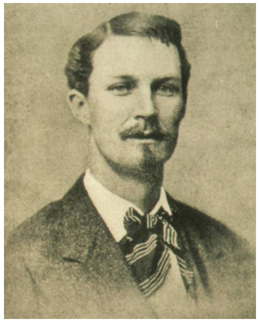
- Ritter: 15K tokens, PTB, one annotator
- Foster: 14K tokens, PTB, low-noise
- CMU: 39K tokens, custom, narrow tagset



Part-of-speech tagging: issues

Unknown words fall roughly into two categories

- Standard token, non-standard orthography;
  - freinds
  - KHAAAANNNNNN!



- Non-standard token, standard orthography
  - omg + bieber → omb
  - Huntingdon / Huntington

Create the ANNIE POS Tags

- Create an Annotation Set Transfer, add to the application
- Set its run-time parameters as shown:

Runtime Parameters for the "Token AST" Annotation Set Transfer:

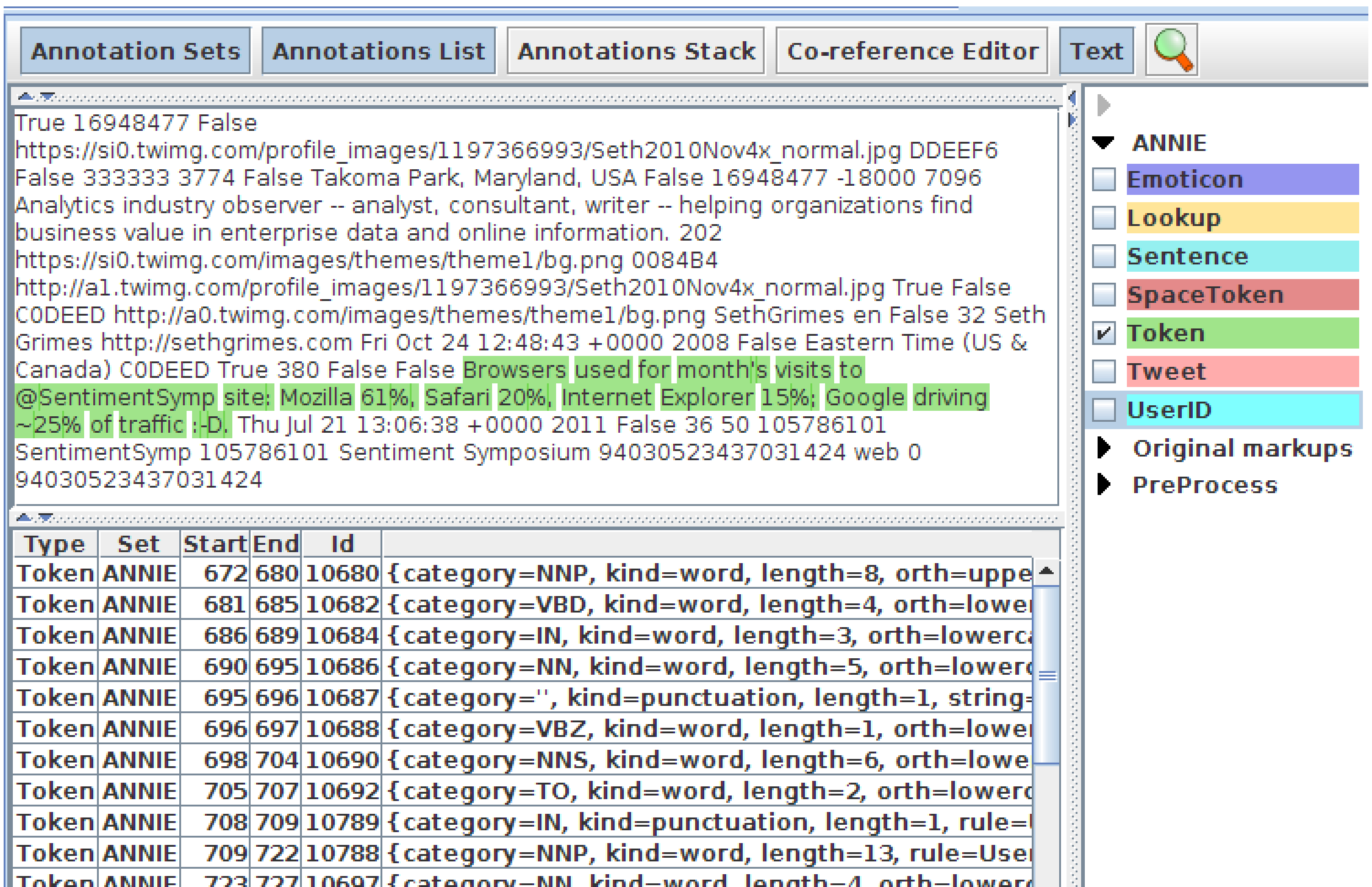
Name	Type	Required	
annotationTypes	ArrayList		[]
copyAnnotations	Boolean	✓	true
inputASName	String		PreProcess
outputASName	String		ANNIE
tagASName	String		PreProcess
textTagName	String		Tweet
transferAllUnlessFound	Boolean	✓	true

- Create an ANNIE POS Tagger with default init parameters
- Add to the application and set run-time params:

Name	Type	Required	
baseSentenceAnnotationType	String	✓	Tweet
baseTokenAnnotationType	String	✓	Token
failOnMissingInputAnnotations	Boolean		true
inputASName	String		ANNIE
outputASName	String		ANNIE
outputAnnotationType	String	✓	Token

Hands-On: ANNIE POS Tags (2)

- Run the application
- Inspect the Token annotations in the ANNIE set



Load & configure the Stanford Tagger

- Create another Annotation Set Transfer, add to the application
- Set its run-time parameters as shown:

Runtime Parameters for the "Tweet POS AST" Annotation Set Transfer:

Name	Type	Required	Value
annotationTypes	ArrayList		[]
copyAnnotations	Boolean	✓	true
inputASName	String		PreProcess
outputASName	String		
tagASName	String		PreProcess
textTagName	String		Tweet
transferAllUnlessFound	Boolean	✓	true

- Load the Stanford_CoreNLP plugin through Plugin Manager
- Create an instance of Stanford POS Tagger with this model:
    - **resources/gate-EN-twitter.model**
- Add to the application at the end
- Run the application

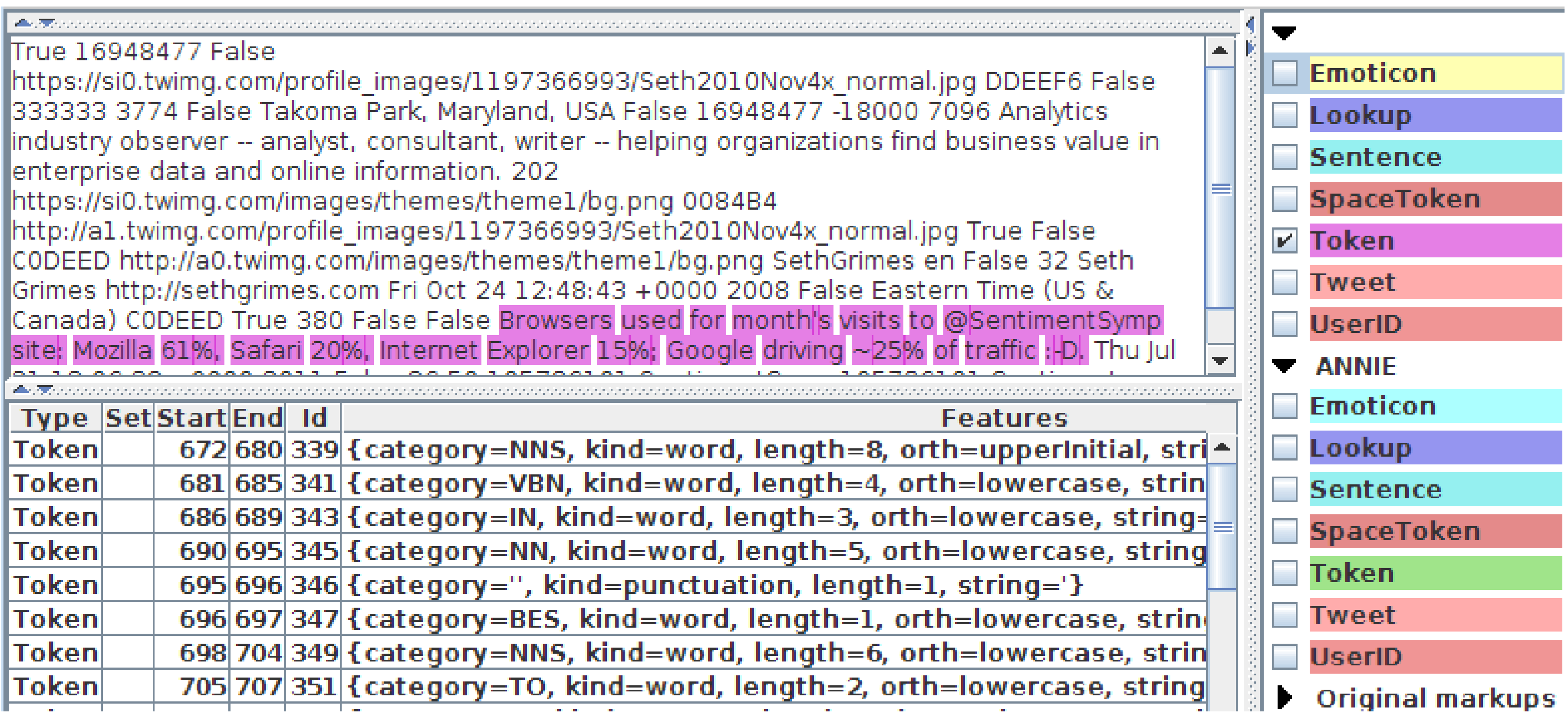
Hands On: App Sanity Check

- By now your tweet processing application should look like this

!	Name	Type
	Document Reset PR_00227	Document Reset PR
	Annotation Set Transfer_0022C	Annotation Set Transfer
	metadata-preprocess	JAPE Transducer
	TwitIE Lang ID	TextCat Language Identification
	ANNIE English Tokeniser_002E7	ANNIE English Tokeniser
	Segment Processing PR_002F1	Segment Processing PR
	Tweet Normaliser_0031A	Tweet Normaliser
	Annotation Set Transfer_00386	Annotation Set Transfer
	ANNIE POS Tagger_00387	ANNIE POS Tagger
	Annotation Set Transfer_003C4	Annotation Set Transfer
	Stanford POS Tagger_003C5	Stanford POS Tagger

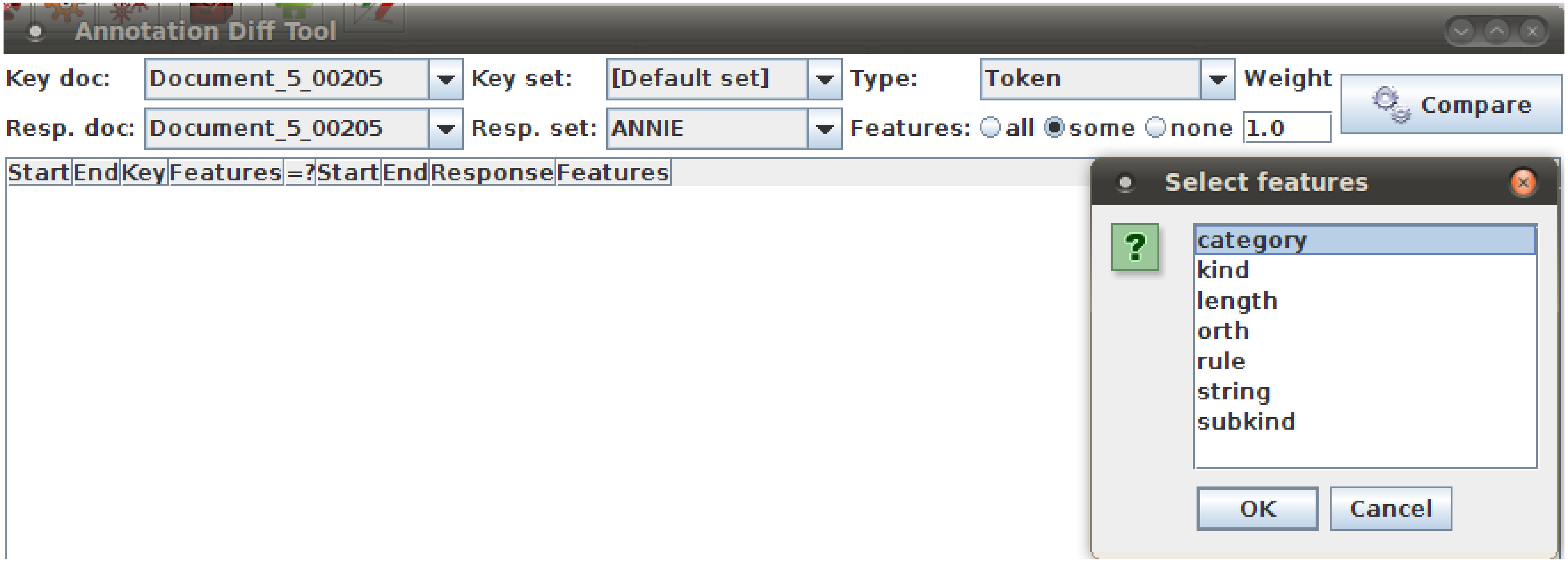
TwitIE POS Tagger Results: Example

- If all has been setup properly, you will get results in 2 sets:
  - ANNIE will have the POS tags from the ANNIE POS Tagger
  - The default set will have those from the TwitIE Tagger



Compare Differences: Annotation Diff

- Click on the Annotation Diff button 
- Select a document from the test corpus (same Key and Resp)
- Key set: [Default set]; Resp. set: ANNIE
- Type: Token; Features: some, then select: category



Compare Differences (2)

- Click on the Compare button
- Inspect the results; repeat for 1-2 more documents
- HINT: Clicking on the Start column will sort tokens by offset

Start	End	Key	Features	=?	Start	End	Response	Features
736	741	Nokia	{category=NNP, kind=...gth=5, string=Nokia}	=	736	741	Nokia	{category=NNP, kind=...gth=5, string=Nokia}
742	747	Posts	{category=VBZ, kind=...gth=5, string=Posts}	=	742	747	Posts	{category=VBZ, kind=...gth=5, string=Posts}
748	752	Huge	{category=JJ, kind=w...ngth=4, string=Huge}	=	748	752	Huge	{category=JJ, kind=w...ngth=4, string=Huge}
753	762	Quarterly	{category=JJ, kind=w...9, string=Quarterly}	=	753	762	Quarterly	{category=JJ, kind=w...9, string=Quarterly}
763	767	Loss	{category=NN, kind=w...ngth=4, string=Loss}	=	763	767	Loss	{category=NN, kind=w...ngth=4, string=Loss}
767	768	,	{string=,, length=1,...tuation, category=,}	=	767	768	,	{string=,, length=1,...tuation, category=,}
769	773	Sees	{category=VBZ, kind=...ngth=4, string=Sees}	<>	769	773	Sees	{category=NNP, kind=...ngth=4, string=Sees}
774	780	Better	{category=NNP, kind=...th=6, string=Better}	=	774	780	Better	{category=NNP, kind=...th=6, string=Better}
781	786	Times	{category=NNP, kind=...gth=5, string=Times}	=	781	786	Times	{category=NNP, kind=...gth=5, string=Times}
787	792	Ahead	{category=NNP, kind=...gth=5, string=Ahead}	=	787	792	Ahead	{category=NNP, kind=...gth=5, string=Ahead}
793	794	-	{category=:, subkind... length=1, string=-}	=	793	794	-	{category=:, subkind... length=1, string=-}
795	819	http://on...	{rule=URL, temp_cate...ed=13, category=URL}	<>	795	819	http://o...	{rule=URL, temp_cate...ced=13, category=CD}

- We are still improving the tweet POS model, but major improvements make it current state-of-the-art
- Beats Ritter (2011); uses a grown-up tag set (cf. Gimpel, 2011)

Tackling the problems: Unseen words in tweets

Majority of non-standard orthographies can be corrected with a gazetteer: typical Pareto

- Vids → videos
- cussin → cursing
- hella → very

No need to bother with e.g. Brown clustering

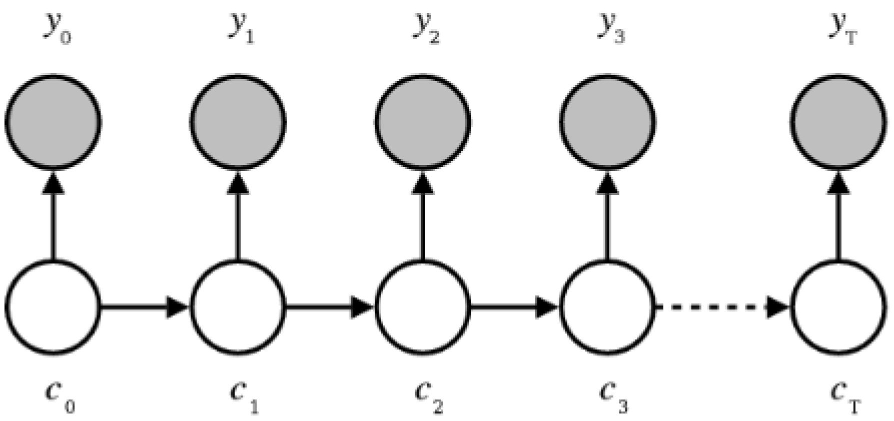
361 entries give 2.3% token error reduction

Tweet “sentence” *“structure”*

Tweets contain some constrained-form tokens

Links, hashtags, user mentions, some smileys

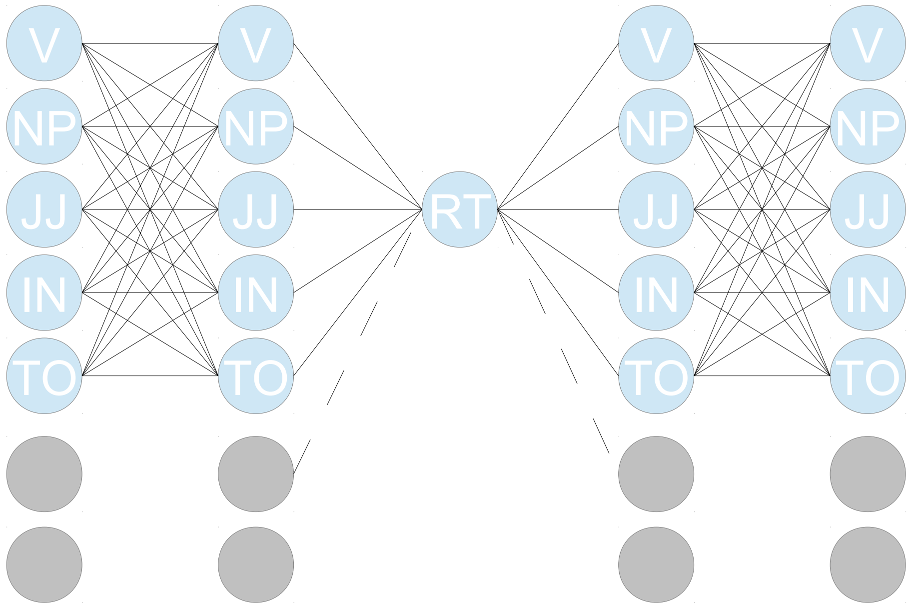
We can fix the label for these tokens



Knowing $P(c_i)$ constrains both $P(c_{i-1}|c_i)$ and $P(c_{i+1}|c_i)$

Tweet “sentence” *“structure”*

This allows us to prune the transition graph of labels in the sequence:



Because the graph is read in both directions, fixing the value of any label point impacts whole tweet

Setting label priors reduces token error **5.03%**

Part-of-speech tagging: solutions

Not much training data is available, and it is expensive to create

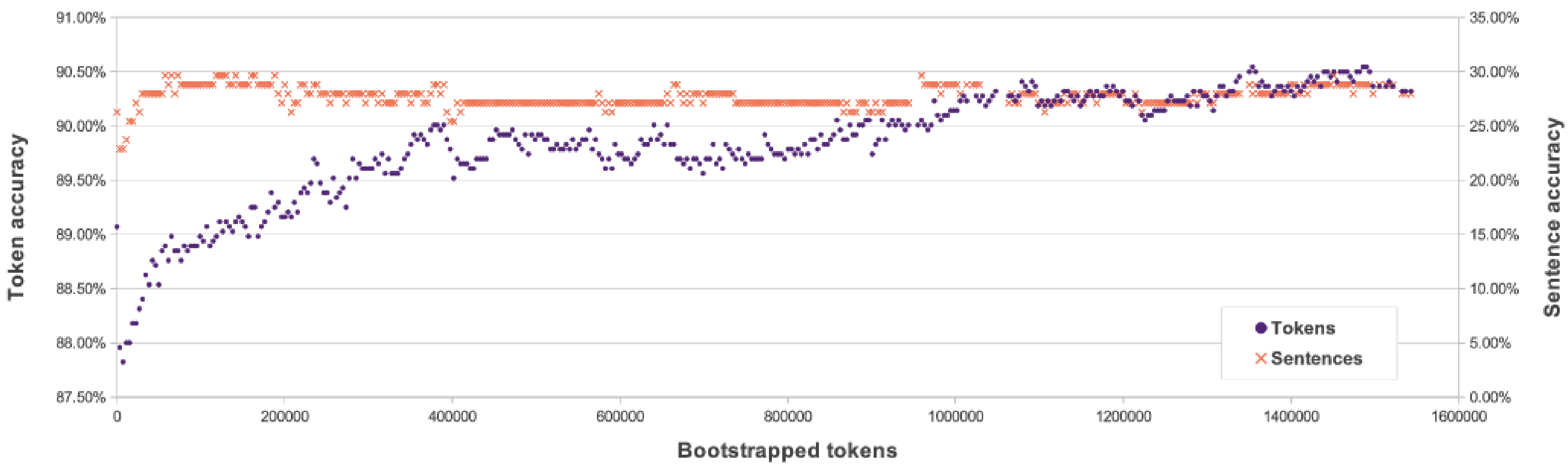
- Plenty of unlabelled data available – enables e.g. bootstrapping
- Existing taggers algorithmically different, and use different tagsets with differening specificity
  - CMU tag **R** (adverb) → PTB (**WRB**,**RB**,**RBR**,**RBS**)
  - CMU tag **!** (interjection) → PTB (**UH**)

Part-of-speech tagging: solutions

Label unlabelled data with taggers and accept tweets where tagger votes never conflict

- Lebron_^ + Lebron_NNP → OK, Lebron_NNP
- books_N + books_VBZ → Fail, reject whole tweet

Token accuracy: 88.7% sentence accuracy: 20.3%



Part-of-speech tagging: solutions

Gimpel et al. (2011) adopt a holistic approach to PoS tagging

A tagset is created that adapts to the tokenisation issues we saw

- No splitting contractions; instead, combined forms added. {nn, nnp} x {vb, pos}
- New tags for twitter phenomena (#, @, ~ for RT, U for URL) and emoticons (E)
- Choose to annotate mid-sentence hashtags as other parts of speech
- Leads to new corpus, tokenised and tagged: 39K tokens, 0.92 IAA

Part-of-speech tagging: solutions

Twitter-specific features used with CRF

- Orthographic, detecting fixed-format tokens
- Frequently-capitalised tokens are collected, to overcome capitalisation inconsistency
- Prior tag distribution taken from PTB (including Brown) as soft prior
- Distributional similarity taken from 1.9M unlabelled tweets, looking one ahead & behind
- Phonetic representations are taken using metaphone, and compared with tag distributions in PTB of words sharing the metaphone key

Owoputi et al. (2013) extend using word clusters, proper name gazetteers, and regularisation

Final accuracy: 93.2% token-level, ~22% sentence-level

Part-of-speech tagging: solutions

Ritter et al. (2011) adapt to twitter by looking beyond newswire and modelling lexical variation

Extra resources include adapting standards to the genre and finding more & better data

- Extension of PTB tagset, with HT, USR, RT, and URL
- Inclusion of an IRC dataset (online chat; assumed similar to twitter; source of hashtag)
- Creation of a new Twitter corpus – 15K tokens, single annotator

Part-of-speech tagging: solutions

Non-standard spelling, through error or intent, is often observed in twitter – but not newswire

- Model words using Brown clustering and word representations (Turian 2010)
- Input dataset of 52M tweets as distributional data
- Use clustering at 4, 8 and 12 bits; effective at capturing lexical variations
  - E.g. cluster for “tomorrow”: 2m, 2ma, 2mar, 2mara, 2maro, 2marrow, 2mor, 2mora, 2moro, 2morow, 2morr, 2morro, 2morrow, 2moz, 2mr, 2mro, 2mrrw, 2mrw, 2mw, tmmrw, tmo, tmoro, tmorrow, tmoz, tmr, tmro, tmrow, tmrrow, tmrrw, tmrw, tmrww, tmw, tomaro, tomarow, tomarro, tomarrow, tomm, tommarow, tommarrow, tommoro, tommorow, tommorrow, tommorw, tommrow, tomo, tomolo, tomoro, tomorow, tomorro, tomorrw, tomoz, tomrw, tomz

Data and features used to train CRF. Reaches 41% token error reduction over Stanford tagger.

PoS tagging: where next?

Better handling of case problems (uppsala_NN, Wine_NNP)

Better handling of hashtags

- I'm stressed at 9am, shopping on my lunch break... can't deal w/ this today. #retailtherapy
- I'm so #bored today

More data – bootstrapped

More data – part-bootstrapped (e.g. CMU GS)

More data – human annotated

Parsing...

Training a PoS tagger

- You can use models for the Stanford tagger
- Here we will learn from some corpora
- Tag inventory defined by training data
- Training custom models allows:
  - Control over features and feature extraction
  - Inclusion of in-genre data
  - Adaptation to new languages

- The models created can be re-used only with the same version of the Stanford framework; we use v. 3.3.1.

Training a PoS tagger

- Corpora take the following format:
  - One sentence per line
  - Labelled tokens separated by spaces
  - In the format TOKEN_LABEL

  when_WRB i_PRP compliment_VBP her_PRP she_PRP wo_MD n't_RB believe_VB mee_PRP

- Training is controlled via a .props file, with various attributes
  - trainFile=file1;file2;file3
  - arch=left3words,generic,naacl2003unknowns
  - learnClosedClassTags
  - minFeatureThresh
  - RareWordThresh
  - ...and many more

  Reference: http://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/tagger/maxent/ExtractorFrames.html

Training a PoS tagger

We'll measure the impact of in-genre data and modelling unknown words. This section means you need Java on your machine.

- Start a new **Corpus pipeline**;
- Create and add a new: **Document Reset;** an **ANNIE English Tokeniser;** an **ANNIE Sentence Splitter;** and a **Stanford PoS Tagger** with default model

- Create a new corpus for POS tagging and save it to DS
- Load documents from XML:
  - File **ritter-twitter.gate.xml** , **doc_root** separator, mimetype **text/xml**

- Run the pipeline using this model
- Use the Annotation Diff tool to compare results to the Original Markup annotation – how does it do?

Next, get corpus performance figures using Corpus QA

- Double-click on the corpus
- Use the “Corpus Quality Assurance” tool
- Compare token annotations and the “category” feature

Training a PoS tagger

- Let's train a tagger model based on newswire

- In the “pos” directory of the handout, you will find a prop file, a README, and some small corpora
- Edit the propfile to:
    - Use just the WSJ data (“trainFile” parameter)
    - Write to model called wsj.model (“model” parameter)

- Save this propfile as wsj.props
- Copy the command for training a tagger model from the README, and using wsj.props, train a new model
    - You might need “Command prompt” or “Terminal” to do this
    - It'll take two to ten minutes, depending on your computer
    - Browse the reference guide for this software if you're bored!

- In GATE, create a new PR that uses this new model file
- Replace the old Stanford PoS Tagger PR with the one using your own model
- You should see similar results to the default english-left3words model, though we used much less training data.

Training a PoS tagger

- Let's change the corpus to a twitter one: twitter.stanford
- Take a look at the data if you like (plain text)

- Copy wsj.props to twitter.props
- Edit the model and trainFile parameters so we're using the twitter dataset file instead and writing to twitter.model
- Learn the model, and run it in your pipeline

- Try the Corpus QA: how good is the performance?

- Next, try using both the WSJ and Twitter training data
- This is done with the trainFile param, and semicolons between filenames
- Output to wsj-and-twitter.model

- What difference is there to performance?

Training a PoS tagger

- Combining twitter and newswire text gives best performance
- There are still lots of problems

- We know that social media text is noisy. What if we added features to our model for handling noise?

- In your propfile for wsj-twitter, add the following options to the list in the “arch” param:
  - `naacl2003unknowns`
  - `lnaacl2003unknowns`

- Re-build the model
- Put it in your Twitter PoS PR, and run it
- What does corpus QA think now?

- State-of-the-art is 91% tag accuracy in tweets.
- How does this model perform?
- What else could help?