Distributional lexical semantics (I)

CS 690N, Spring 2018

Advanced Natural Language Processing

http://people.cs.umass.edu/~brenocon/anlp2018/

Brendan O'Connor

College of Information and Computer Sciences

University of Massachusetts Amherst

- Lexical resources are good (e.g. WordNet or name lists)
  - But hand-built ones are hard to create/maintain
- "You shall know a word by the company it keeps (Firth, 1957)

(15.1) *A bottle of ________ is on the table.*

(15.2) *Everybody likes ________.*

(15.3) *Don't have ________ before you drive.*

(15.4) *We make ________ out of corn.*

Word-context matrix (derived from counts; e.g. Pos. PMI...)

	C1	C2	C3	C4	...
tezgüino	1	1	1	1	
loud	0	0	0	0	
motor oil	1	0	0	1	
tortillas	0	1	0	1	
choices	0	1	0	0	
wine	1	1	1	1	

- Context representation
  - Word windows
  - Directionality
  - Syntactic context
- What to do with context counts
  - Sparse, original form (rescaled: PPMI)
  - Dimension reduction: SVD or log-bilinear models (word2vec)
  - Model-based approaches
    - Markov-ish: Saul&Pereira
    - HMM-ish: Brown clustering

Features from unsup. learning

- Generative models allow unsupervised learning;
  can use as features
  - Baum-Welch algo.: unsup. HMM via EM
    (its categories seem to disagree with parts of speech)
- Brown word clustering *[Brown et al. 1992]*
  - HMM + one-class constraint: Every word belongs to only one class
    (bad assumption, but better than alternative;
    [Blunsom et al. 2011])
  - Why HMM learning looks like distributional clustering [whiteboard]
  - Agglomerative clustering: yields binary tree over clusters
    - alternative: agglom cluster in word embedding space?
  - Compare to using word embeddings as linear features:
    allows multiresolutional generalizations
  - Very useful as CRF features for POS, NER
    [Turian et al. 2010, Derczynski et al. 2015]
    http://www.derczynski.com/sheffield/brown-tuning/

Word clusters as features

- Labeled data is small and sparse. Lexical generalization via induced word classes.
  - Both embeddings and clusters can be used as features!
- Examples from Twitter, for POS tagging
  - Unlabeled: 56 M tweets, 847 M tokens
  - Labeled: 2374 tweets, 34k tokens
- 1000 clusters over 217k word types
  - Preprocessing: discard words that occur < 40 times

[Owoputi et al. 2013]

http://www.ark.cs.cmu.edu/TweetNLP/cluster_viewer.html

What does it learn?

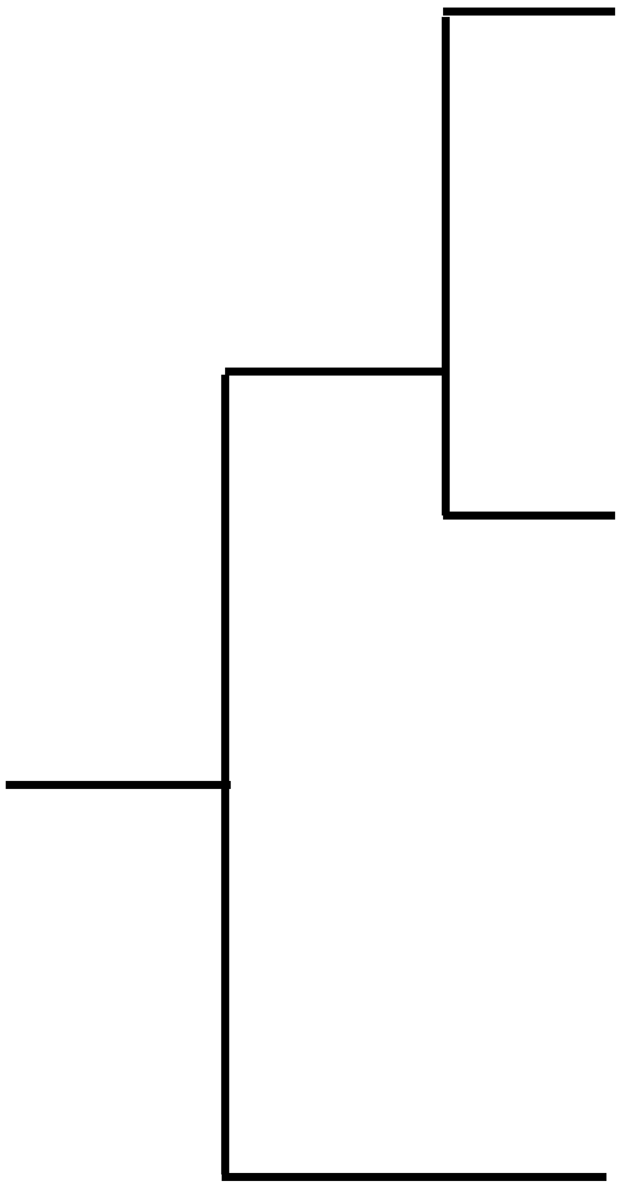
- Orthographic normalizations

so s0 -so so- $o /so //so

soo sooo soooo sooooo soooooo sooooooo soooooooo sooooooooo soooooooooo sooooooooooo soooooooooooo sooooooooooooo soso soooooooooooooo sooooooooooooooo soooooooooooooooo sososo superrr sooooooooooooooooo ssooo so0o superrrr so0 soooooooooooooooooo sosososo sooooooooooooooooooo ssoo sssooo soooooooooooooooooooo #too s0o ssoooo s00 sooooooooooooooooooooo so0o0o sososososo soooooooooooooooooooooo sssoooo ssooooo superrrrr very2 s000 sooooooooooooooooooooooo soooooooooooooooooooooooo sooooooooooooooooooooooooo _so_ soooooooooooooooooooooooooo /so/ sssooooo sosososososo

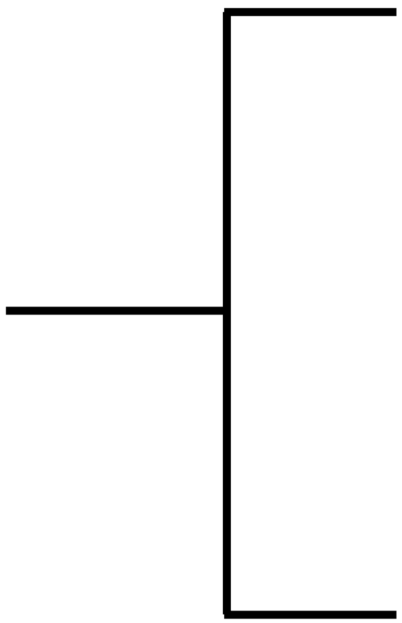
- suggests joint model for morphology/spelling

- Emoticons etc.
(Clusters/tagger useful for sentiment analysis: NRC-Canada SemEval 2013, 2014)



:d ^^ =d *-* :-d \o/ :dd \m/ 8d *--* *_* u.u :ddd ;;) *.* o/ ;3 =))) *---* \(´▽`)/ n_n b-) (^_^) ^o^ :dddd ;dd *__* :)))))) *----* d/ \o \: =dd n.n -q *___* :33 :ddddd :od -n *-----* xddddd <URL-crunchyroll.com> ^^v (x \= =:) *------* \0/ (˘_˘")
;) :p :-) xd ;-) ;d (; :3 ;p =p :-p =)) ;] xdd 😏 #gno xddd >:) ;-p >:d □ 8-) □ 😜 □ ;-d □ 😝 [; □ :^) =)))) ;-)) <URL-seesmic.com> :pp :~) x'd :op >:p ;^) >:] =))))) :>) <URL-hstl.co> ;)))) ;~) toort >:3 #eden ;pp
:) (: =) :)) :] ☺ :') =] ^_^ :))) ^.^ [: ;)) 😊 ((: ^__^ (= ^-^ :)))) 😁 👍 □ :-)) 😉 ✌ ^___^ (': :} :))))) □ 😃 👌 □ □ 😌 :") :]] □ =]] 😄 □ □ ü ;))) [= (-: ^____^ ;') :-))) (((:
:o o_o « o.o xp ;o ._. t.t t_t #wtf #lol o: x_x =o 0_o dx o_0 :-o ¬¬ --" 0_0 o__o »» u_u #help --' =3 (-_-) -) #confused □ #omg ¬_¬ t^t otl #igetthatalot 😱 xdddd o___o @@ cx t__t d8 □ :{ t___t ---------- #whodoesthat e_e :oo
:(:/ -_- -.- :-(:'(d: :l :s -__- =(=/ >.< -___- :-/ </3 :\ -____- ;(/: ☹ :((>_< =[:[#fml 😒 -_____- =\ >:(😔 -,- >.> >:o ;/ 😳 d; .-. -______- >_> :(((-_-" =s □ ;_; #ugh :-\ =.= □ -_______-
x xx xxx xxxx xxxxx qt xox xxxxxx xxxxxxx xxxxxxxx #pawpawty xxxxxxxxx xxxxxxxxxx #1dfamily #frys #1dqa xxxxxxxxxxx #askliam #dcth xxxxxxxxxxxx #askniall *rt #jbinpoland xxxxxxxxxxxxx #askharry x-x #wiimoms xxxxxxxxxxxxxx oxox #wlf #nipclub +) 1dhq xxxxxxxxxxxxxxx #20peopleilove <URL-paidmodels.com> yart #jedreply #elevensestime <URL-shrtn.us> #askzayn xxxxxxxxxxxxxxxx #wineparty +9 #amwritingparty #tweepletuesday #soumanodomano <URL-today.com> #twfanfriday 22h22
<3 ♥ xoxo <33 xo <333 ❤ ♡ #love s2 <URL-twitition.com> #neversaynever <3333 #swag x3 #believe #100factsaboutme ♥♥ 😘 <3<3 <33333 #blessed xoxoxo 😍 #muchlove #salute xoxox ♥♥♥ #excited ☀ □ #happy #leggo #cantwait <3<3<3 #loveit <333333 #please #dailytweet #thanks 🙏 (˘‿˘) 💜 #yay #thankyou #loveyou {} ε˘) #nsn #iloveyou

(Immediate?) future auxiliaries

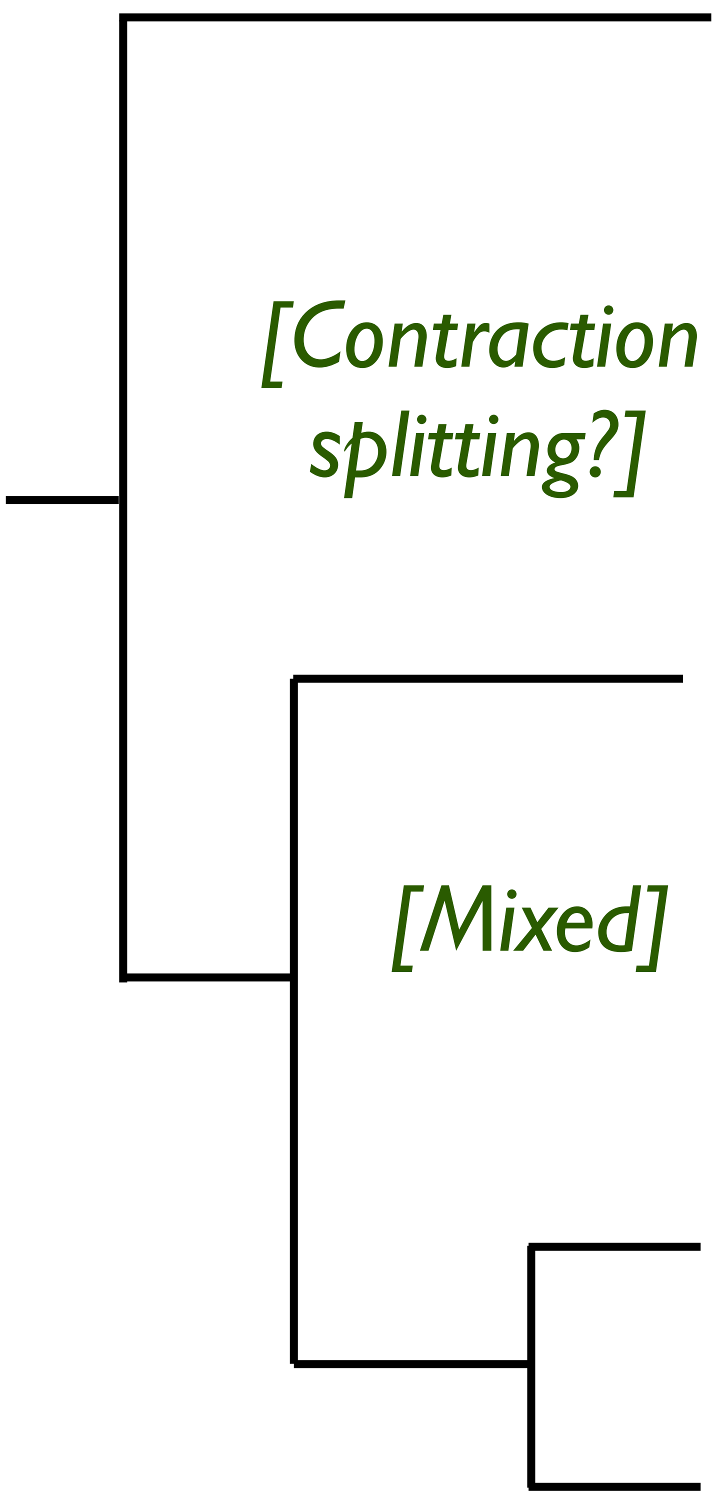


gonna gunna gona gna guna gnna ganna qonna gonnna gana qunna gonne goona gonnaa g0nna goina gonnah goingto gunnah gonaa gonan gunnna going2 gonnnna gunnaa gonny gunaa quna goonna qona gonns goinna gonnae qnna gonnaaa gnaa

tryna gon finna bouta trynna boutta gne fina gonn tryina fenna qone trynaa qon boutaa funna finnah bouda boutah abouta fena bouttah boudda trinna qne finnaa fitna aboutta goin2 bout2 finnna trynah finaa ginna bouttaa fna try'na g0n trynn tyrna trna bouto finsta fnna tranna finta tryinna finnuh tryingto boutto

- finna ~ “fixing to”
- tryna ~ “trying to”
- bouta ~ “about to”

Subject-AuxVerb constructs



i'd you'd we'd he'd they'd she'd who'd i’d u'd youd you’d iwould theyd icould we’d i`d #whydopeople he’d i´d #iusedto they’d i'ld she’d #iwantsomeonewhowill i'de imust a:i'd you`d yu'd icud l'd

ill ima imma i'ma i'mma ican iwanna umma imaa #imthetypeto iwill amma #menshouldnever igotta #whywouldyou #iwishicould #sometimesyouhaveto #thoushallnot #ihatewhenpeople illl #thingspeopleshouldnotdo #howdareyou #thingsgirlswantboystodo im'a #womenshouldnever #thingsblackgirlsdo immma iima #ireallyhatewhenpeople ishould #thingspeopleshouldntdo #irefuseto itl #howtospoilahoodrat iwont imight #thingsweusedtodoaskids ineeda #thingswhitepeopledo we'l #whycantyoujust #whydogirls #everymanshouldknowhowto #ushouldnt #howtopissyourgirloff #amanshouldnot #uwannaimpressme #realfriendsdont immaa #ilovewhenyou

you'll we'll it'll he'll they'll she'll it'd that'll u'll that'd youll ull you’ll itll there'll we’ll itd there'd theyll this'll thatd thatll they’ll didja he’ll it’ll yu'll she’ll youl you`ll you'l you´ll yull u'l it'l we´ll we`ll didya that’ll it’d he'l shit'll they'l theyl she'l everything'll he`ll things'll u’ll this'd

i'll i’ll i'l i`ll i´ll i'lll l'll i\'ll i"ll -i'll /must @pretweeting she`ll

Word clusters as features

ikr	smh	he	asked	fir	yo	last
!	G	O	V	P	D	A
name	so	he	can	add	u	on
N	P	O	V	V	O	P
fb	lololol					
^	!					

w fo fa fr fro ov fer **fir** whit abou aft serie fore fah fuh w/her w/that fron isn agains — "non-standard prepositions"

yeah yea nah naw yeahh nooo yeh noo noooo yeaa **ikr** nvm yeahhh nahh nooooo — "interjections"

facebook **fb** itunes myspace skype ebay tumblr bbm flickr aim msn netflix pandora — "online service names"

smh jk #fail #random #fact smfh #smh #winning #realtalk smdh #dead #justsaying — "hashtag-y interjections"??

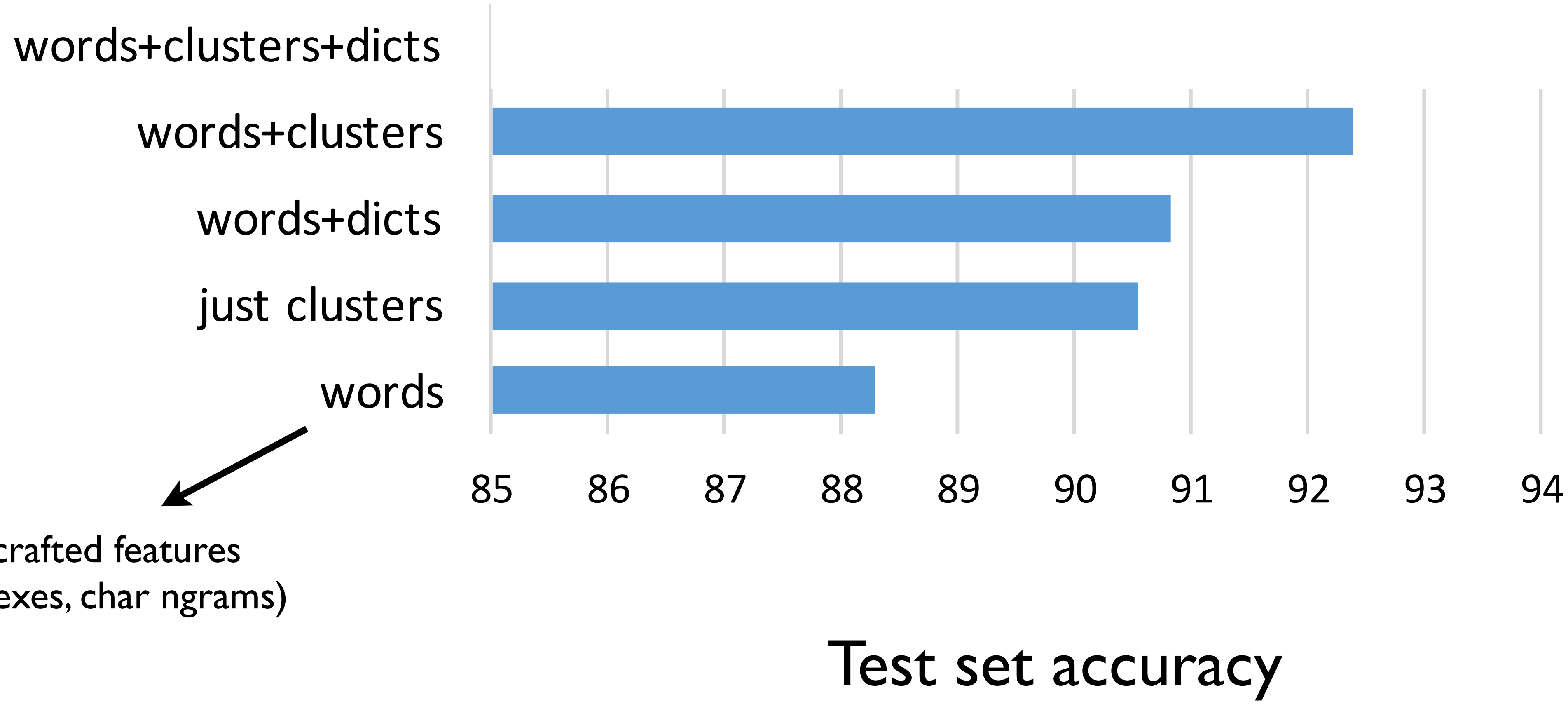
Highest-weighted POS–treenode features

hierarchical structure generalizes nicely.

Cluster prefix	Tag	Types	Most common word in each cluster with prefix
11101010*	!	8160	lol lmao haha yes yea oh omg aww ah btw wow thanks sorry congrats welcome yay ha hey goodnight hi dear please huh wtf exactly idk bless whatever well ok
11000*	L	428	i'm im you're we're he's there's its it's
1110101100*	E	2798	x <3 :d :p :) :o :/
111110*	A	6510	young sexy hot slow dark low interesting easy important safe perfect special different random short quick bad crazy serious stupid weird lucky sad
1101*	D	378	the da my your ur our their his
01*	V	29267	do did kno know care mean hurts hurt say realize believe worry understand forget agree remember love miss hate think thought knew hope wish guess bet have
11101*	O	899	you yall u it mine everything nothing something anyone someone everyone nobody
100110*	&	103	or n & and

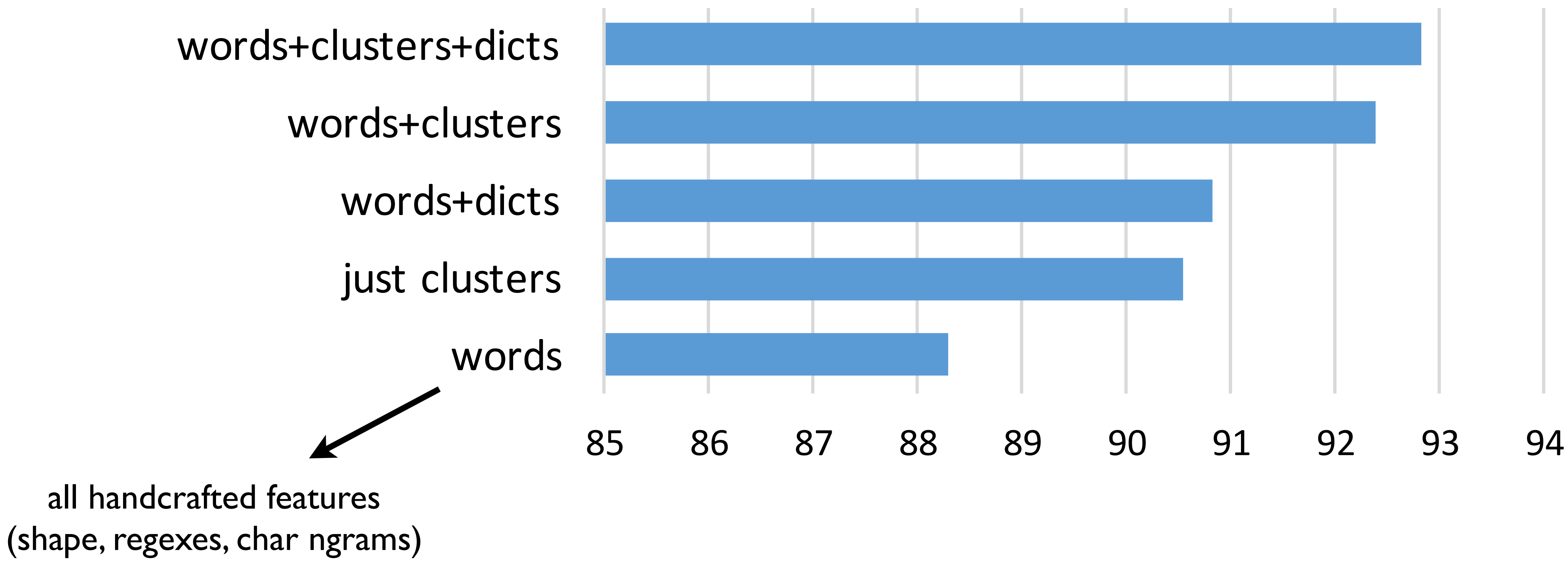
Clusters help POS tagging

Clusters help POS tagging



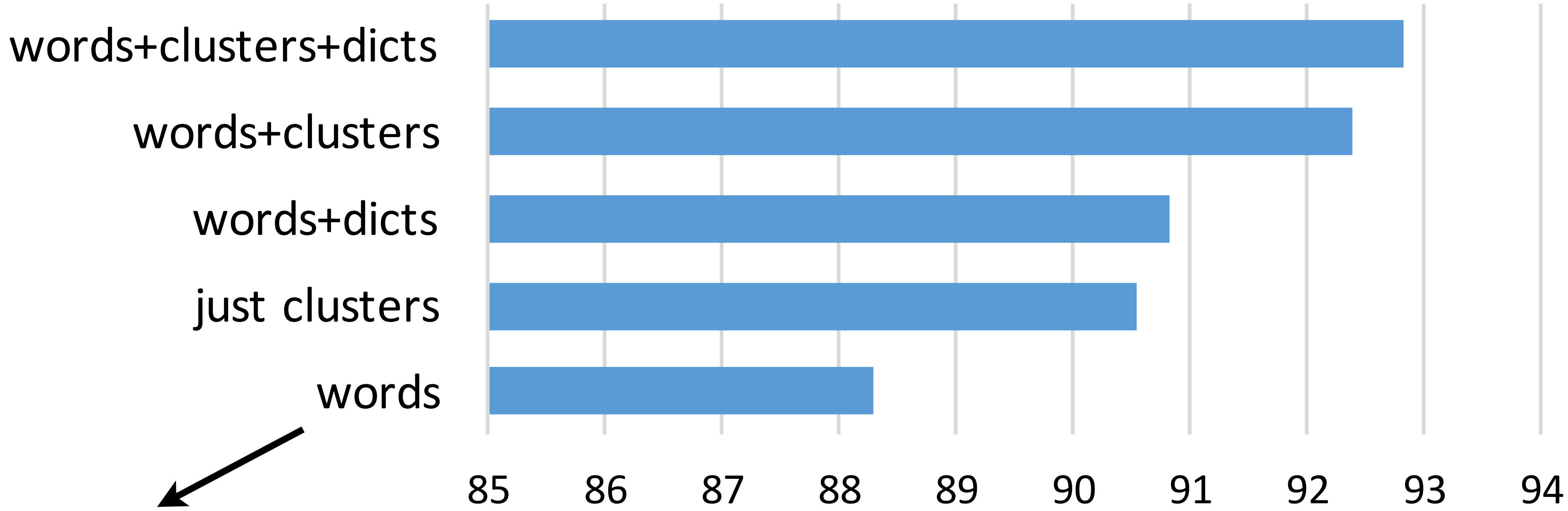
all handcrafted features
(shape, regexes, char ngrams)

Clusters help POS tagging



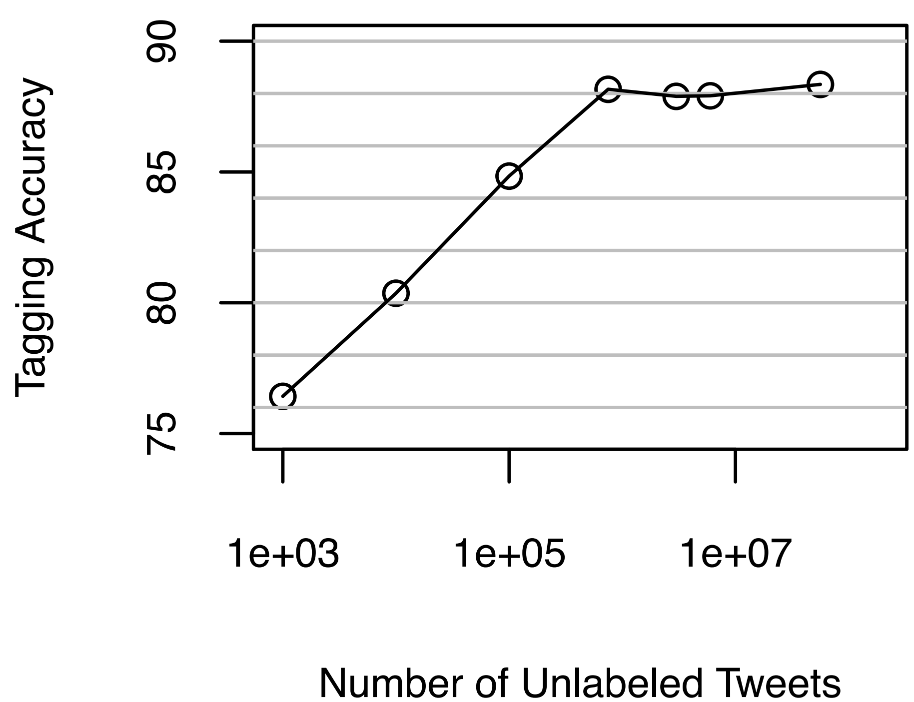
Test set accuracy

Clusters help POS tagging



all handcrafted features
(shape, regexes, char ngrams)

Test set accuracy



Dev set accuracy
using only clusters as features

Direct context approach

- Goal: pairwise similarities
  - Rank nearest neighbors
  - Agglomerative clustering
- 1. Context representation
- 2. Rescaling (if want real-valued). PPMI is popular
  - PPMI(w,c) = max(0, PMI(w,c))
- 3. Similarity metric
  - Cosine similarity (and other L2-ish metrics)
  - Jaccard or Dice similarity (boolean-valued...)
  - Mutual information, etc...

Lin (1998)

- Syntactic contexts (e.g. C -dobj> W)
- Direct context similarity

Nouns		
Rank	Respective Nearest Neighbors	Similarity
1	earnings profit	0.572525
11	plan proposal	0.47475
21	employee worker	0.413936
31	battle fight	0.389776
41	airline carrier	0.370589
51	share stock	0.351294
61	rumor speculation	0.327266
71	outlay spending	0.320535
81	accident incident	0.310121
91	facility plant	0.284845
101	charge count	0.278339
111	baby infant	0.268093
121	actor actress	0.255098
131	chance likelihood	0.248942
141	catastrophe disaster	0.241986
151	fine penalty	0.237606
161	legislature parliament	0.231528
171	oil petroleum	0.227277
181	strength weakness	0.218027
191	radio television	0.215043
201	coupe sedan	0.209631

Adjective/Adverbs		
Rank	Respective Nearest Neighbors	Similarity
1	high low	0.580408
11	bad good	0.376744
21	extremely very	0.357606
31	deteriorating improving	0.332664
41	alleged suspected	0.317163
51	clerical salaried	0.305448
61	often sometimes	0.281444
71	bleak gloomy	0.275557
81	adequate inadequate	0.263136
91	affiliated merged	0.257666
101	stormy turbulent	0.252846
111	paramilitary uniformed	0.246638
121	sharp steep	0.240788
131	communist leftist	0.232518
141	indoor outdoor	0.224183
151	changed changing	0.219697
161	defensive offensive	0.211062
171	sad tragic	0.206688
181	enormously tremendously	0.199936
191	defective faulty	0.193863
201	concerned worried	0.186899

Figure 15.3: Similar word pairs from the clustering method of Lin (1998)

brief (noun): affidavit 0.13, petition 0.05, memorandum 0.05, motion 0.05, lawsuit 0.05, deposition 0.05, slight 0.05, prospectus 0.04, document 0.04 paper 0.04, ...

brief (verb): tell 0.09, urge 0.07, ask 0.07, meet 0.06, appoint 0.06, elect 0.05, name 0.05, empower 0.05, summon 0.05, overrule 0.04, ...

brief (adjective): lengthy 0.13, short 0.12, recent 0.09, prolonged 0.09, long 0.09, extended 0.09, daylong 0.08, scheduled 0.08, stormy 0.07, planned 0.06, ...

- Advantage of syntactic preprocessing: delineate syntactic-level word senses
  Lin (1998)

Latent space approach

- Reduce dimensionality
  - e.g. SVD. Prediction interp: from reduced dim space, best L2-minimizing predictions?
  - *or:* gradient learning for bilinear model
- Typically better than original context space
  - Denoising: low count contexts too noisy?
  - Generalization?
  - Computationally: smaller size (fit on phone...)

Learning Embeddings by Preserving Similarity

- Given long, sparse context cooccurrence vectors V_i and V_j
- Goal: Choose Embeddings E_i and E_j such that similarity is approximately preserved

$$V_i^\top V_j \approx E_i^\top E_j$$

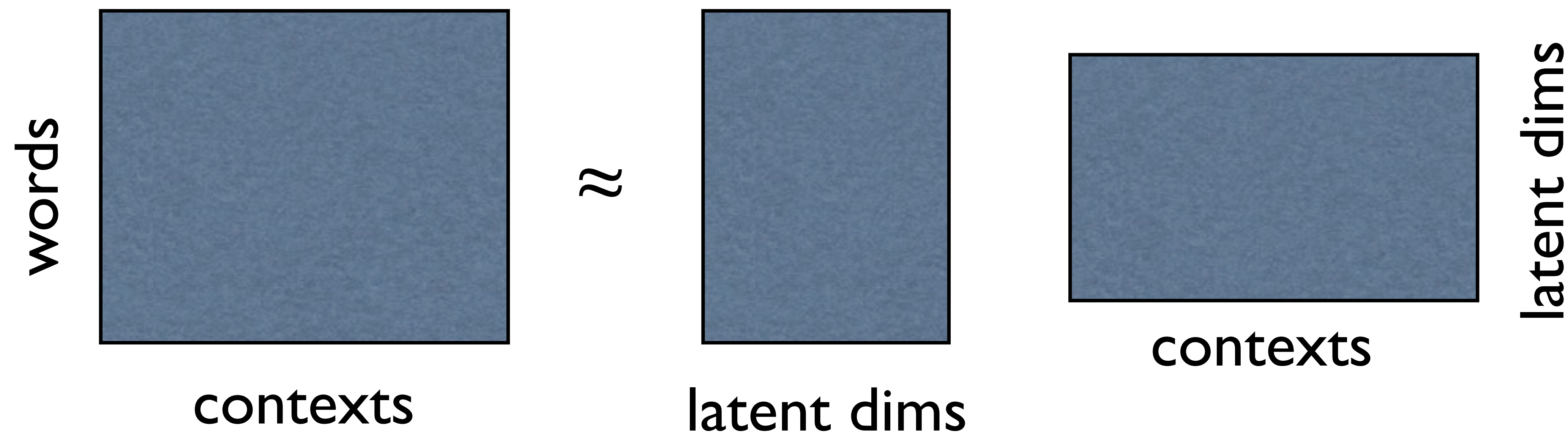
For all words jointly?

Use eigendecomposition / singular value decomposition / matrix factorization

Matrix factorization

V ≈ E B

(counts) (embeddings)



Reconstruct the co-occurrence matrix

$$V_{i,c} \approx \sum_k E_{i,k} B_{k,c}$$

Singular Value Decomposition learns E,B
(or other matrix factorization techniques)

Preserve pairwise distances between words i, j

$$V_i^\mathsf{T} V_j \approx E_i^\mathsf{T} E_j$$

Eigen Decomposition learns E

Skip-gram model

$$u_\theta(w,c) = \exp\left(\boldsymbol{a}_w^\top \boldsymbol{b}_c\right)$$

$$J = \frac{1}{M} \sum_m \sum_{-c \le j \le c, j \ne 0} \log \mathrm{p}(w_{m+j} \mid w_m)$$

$$\begin{aligned}\mathrm{p}(w_{m+j} \mid w_m) &= \frac{u_\theta(w_{m+j}, w_m)}{\sum_{w'} u_\theta(w', w_m)} \\ &= \frac{u_\theta(w_{m+j}, w_m)}{Z(w_m)}\end{aligned}$$

- *[Mikolov et al. 2013]*
- In word2vec. Learning: SGD under a contrastive sampling approximation of the objective
- Levy and Goldberg: mathematically similar to factorizing a PMI(w,c) matrix; advantage is streaming, etc. (though see Arora et al.'s followups...)
- Practically: very fast open-source implementation
- Variations: enrich contexts

- “Distributional / Word Embedding” models
  - Typically, they learn embeddings to be good at word-context factorization,
    which seems to often give useful embeddings
- Pre-trained embeddings resources
  - *GLOVE*, *word2vec*, etc.
  - Make sure it’s trained on a corpus sufficiently similar to what you care about!
- How to use?
  - Fixed (or initializations) for word embedding model parameters
  - Similarity lookups

Extensions

- Alternative: Task-specific embeddings (always better...)
- Multilingual embeddings
- Better contexts: direction, syntax, morphology / characters...
- Phrases and meaning composition
  - vector(hardly awesome) = g(vector(hardly), vector(awesome))