Tasks, Prioritized here & categorized below

* Add to notes below the features used for sentiment task
* Ability to turn on/off features for X-validation tests (good for paper) –see below
* fix last XML reader bugs (see detailed notes)
* binary features (needed for Mallet in any case, & might help w/ others) – see below
* Mallet – Hao to run once coded? And he can help create training / test files
* Usefulness of full sentence sentiment for aspect task (from gold sentiments)
* See if IO might work vs IOB (how many w/ nothing in between?)
* Expand sentiment lexicon

**Feature improvements, either task**

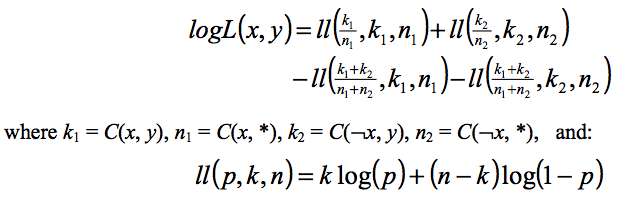
* Either explicitly use binary features for each feature, with (f, v)= 1 if f(t) = v (eg, if word at prevpos is A, use prevpos\_A=1 as feature and all other prevpos\_X features =0)
  + Need to keep vocab around for unk words at test time – add some unks at train time so they’re in features!
* vocabulary/unk features –
  + how many of the recall errors are due to unknown words? Need to do some error analysis
  + Experiment more with unk word handling in general – didn’t do well but that was before Unicode fixes of last third of March
* Would stemming help?
* Test usefulness of full sentence sentiment as a feature by using the majority of the gold standard aspect term polarities as the sentiment – distinguish between balanced pro/con and neutral; Are the neutral labels corresponding to no aspect terms?
  + If good: Figure out how to use the better lexicon in the baseline sentence sentiment clf
  + Learn other indicators that flip the dominant sentiment (like if one aspect term following another has a flipped label)
* add ability to turn on & off different subsets of the features easily
  + create new dictionaries by copying then removing some key
  + wrapper around classifier (fit/train & predict/classify) that only uses the proper keys in the passed example
* Nltk may allow something like tags since determiner or since sentiment word, or words within punctuation boundaries. See the book near end of chunking chapter (tags\_since\_det?)
* expand n-gram window
* character n-grams (could work for sentiment too)
* use the classifier from the other domain as additional evidence (for all the tasks but starting with aspect term extraction)
* add the actual sentiment indicator words (those from the lexicon) as features, try different approaches:
  + closest
  + all
  + if in same (hi-level) NP
  + the actual word, its polarity, or just the presence of such a word
* Hao: add a feature that indicates whether the word is one of the category words (food, etc.), and later extend to looking for its wordnet synset,
  + if parsing indicate whether the current word is in the same clause as a category word

**Other clfs to try, either task**

* Mallet
* For sklearn clfs, change unseen words at test time handling – have to vectorize both the training and test set at the same time. Get around by looking up original features and only using those at test time
* RF
* SVM
  + Problem: very slow training as I’ve now designed it!!
  + Need to create binary classifiers
    - Pairwise (K X (K-1))/2 classifiers
  + See this paper for more: <http://acl.ldc.upenn.edu/N/N01/N01-1025.pdf?origin=publication_detail>
* HMM: would it help to add more features?

**Aspect extraction-specific**

* Figure out how to incorporate PMI & log-likelihood into decision of whether to “combine” two words as part of an aspect phrase. (this paper suggest using a minimum for PMI as a threshold then log likelihood test: (where c(x,y) is count and \* is wildcard) <http://webdocs.cs.ualberta.ca/~lindek/papers/ai01.pdf>



* use a heuristic such as has there been a punctuation “and” “or” or “but” between the current term and the prior term?
* Could try just IO tagging instead of IOB (would be easier only if there are rarely two adjacent aspect terms – could easily test on train data)
* a feature that indicates we’ve seen the word but it’s never been in an aspect term (or the proportion of the time it’s been in an aspect term) – this is only if precision is suffering, which it’s not!
* could also use a NP chunker as additional evidence for aspect term extraction – could boost recall
* use the verb (or adj like “faithful”) & its arguments and only consider args that modify opinionated verbs (or verbs like “said” which denote opinion). Breck etal ’07 take this approach

**Aspect Sentiment specific**

* perhaps divide into two phases: subjective vs objective then pos/neg/mixed
* closest sentiment indicator word and/or its polarity (really want the one closest in a parse) – could chunks be good enough instead of full dep. P\parse?
* extend handling of negation
  + create “words” out of the negation concatenated with previous & next words
  + get (from parser) the word it modifies
* feed sentiment classification back into aspect term extraction – if there is only neutral text then there is no aspect term
* types of patterns
  + the X was Y (steak was delicious)
  + Y X (poor service)
* How to use sentences for which there was no aspect term & thus no sentiment? Words used in a neutral setting, perhaps?
* PMI between sentiment words and heads of aspect phrases?

**Sentence sentiment clf / lexicon**

* If the sentiment lexicon is helping at all, it’s got to be incomplete. Treat modifiers / verbs of aspect terms as new opinion terms?
* Or: Augment the baseline sentiment dictionary from non-aspect terms appearing in the training data (& use the polarity tags to influence their learned polarity)
* Extend with words that occur in similar contexts to base words (but it’s a pretty large lexicon already – not sure in our small corpus if we’d find much, but maybe so since it’s domain specific)

**Performance results summary**

* do some error analysis: features of FPs and FNs, stuck into a dictionary & correlated with where errors were?
* baseline sentiment hand-built classifier based on punctuation boundaries
  + 46.8% on lap, 45.2% if use conjunction as punctuation boundary
  + 48.5% on rest, 45.1 if use conj as punc boundary
* **Aspect extract**
  + Test set recall very low!
    - Laptops: P .75, R: .40, F: .526
    - Rest: P: .78, R: .64, F: .707
  + Restaurant XVal best results (5-fold):
    - **ave Prec: 0.82, Rec: 0.69, F1: 0.75**
    - **Maxent, w/ sentiment lexicons &** 'word': word, 'pos': pos, 'sentiment': sentiment, 'obj': objectivity, 'prevw': prevw, 'prevpos': prevpos, 'prevtag': prevtag, 'prev\_sentiment': prev\_sentiment, 'prev\_obj': prev\_obj, 'nextw': nextw, 'nextpos': nextpos, 'next\_sentiment': next\_sentiment, 'next\_obj': next\_obj}
    - no dep parsing
    - **I can get better recall (.74) but much worse P (.66)** with BO instead of IOB – but this is a bad compare, the eval fn counts all Bs, need to rewrite (laptop was 0.54, Rec: 0.58, F1: 0.56)
  + Laptop XVal best results, same features: **ave Prec: 0.78, Rec: 0.53, F1: 0.63**
* **Aspect sentiment**
  + Restaurant: Maxent, baseline sentiment handbuilt clf, **0.649116159535**
  + Laptop: **0.64797886584**

**Category tasks**

* one classifier per topic/category and mixture of experts approach

**Twitter task**

* Features for twitter task
  + all caps
  + initial cap
  + hashtag (not for aspect task)
  + stemmed vs not
  + stopwords vs not
  + lowercase vs not
  + emoticons as a single feature
  + other normalization (numbers, hashtags)
  + whether a RT or not
  + removing URLs (but keeping them aside for later getting the text in them)
  + replacing @<uname> with just “@”

**Detailed notes**

* CRFs w/Mallet: if using SimpleTagger, each sequence is separated by a blank line (input file to train / test on), and features are binary, so each word is its own feature. Could also call Mallet from NLTK instead of doing it separately, which might be needed for going beyond the SimpleTagger
* Bugs in xml reader:
  + quotes
  + Also doesn’t work if the word ends with a dash

IOB tagging notes:

* In the restaurant training data, there are (might miss a few aspects due to bugs in XML parser as of 3/31)
  + 3041 sentences
  + 3685 aspect terms (ave of 1.2/sentence)
  + 7 occurrences of sequential aspect terms with no intervening tokens (or .18996%)
* laptop training data
  + 3045 sentences
  + 2334 aspect terms (.767/sentence)
  + 1 occurrence of a sequential aspect term
* The “to” position in the xml is the position after the EOW, with punctuation stripped, if any

Sentiment lexicon notes:

* B.Liu’s is just word list for pos & neg
* Wilson etals associates words with pos tags and stemmed versions of the words, so to use that well need stemming

Sentiment sequence classifier (hand-built) as of late March. Makes errors on:

The tech guy then said the service center does not do 1-to-1 exchange and I have to direct my concern to the "sales" team, which is the retail shop which I bought my netbook from. (Negates concern to be positive; the AND should be like a punc boundary there? There are other cases where it shouldn’t be)

Note: due to remaining bugs in XMLReader, (as of time of submission of phase A)

The upper bound on my PR/F for laptops on the training data is :

#of sentences: 3045

#System Aspect Terms=2334

#Gold Aspect Terms=2358

Pre: 0.9888603 (2308/2334)

Rec: 0.9787956 (2308/2358)

F: 0.9838022

And for restaurants:

#of sentences: 3041

#System Aspect Terms=3685

#Gold Aspect Terms=3693

Pre: 0.99674356 (3673/3685)

Rec: 0.9945843 (3673/3693)

F: 0.99566275

Test set sentences

Lap: 800

# of guesses I made: 350

Rest: 800

# of guesses: 934

* The semeval\_base gets P = 0.401709 -- R = 0.381339 -- F1 = 0.391259 (#correct: 188, #retrieved: 468, #relevant: 493) on laptops & P = 0.539118 -- R = 0.514247 -- F1 = 0.526389 (#correct: 379, #retrieved: 703, #relevant: 737) on restaurants

**results , aspect extraction**

* Hmm R/P on 500 sentence subset of laptop:
  + with just pos tags as features: .03 / .5
  + with wd+pos: .08/.75
  + with stemmedWd+pos: .16 / .86
  + just stemmedWd, no POS: .32, .86

3/19, 'word': word, 'pos': pos, 'sentiment': sentiment, 'obj': objectivity,

'prevw': prevw, 'prevpos': prevpos, 'prevtag': prevtag, 'prev\_sentiment': prev\_sentiment, 'prev\_obj': prev\_obj,

'nextw': nextw, 'nextpos': nextpos, 'next\_sentiment': next\_sentiment, 'next\_obj': next\_obj}

After fixing Unicode issues, above features, laptop: ave Prec: 0.77, Rec: 0.53, F1: 0.63

Restaurant: ave Prec: 0.82, Rec: 0.69, F1: 0.75

Now running with turning on stemming:

ave Prec: 0.82, Rec: 0.69, F1: 0.75 (no difference but I believe it kicked in)

* (3/16) our version of P/R/F with maxent and these features: 'word': word, 'pos': pos, 'sentiment': sentiment, 'prevpos': prevpos, 'prevtag': prevtag, 'prev\_sentiment': prev\_sentiment,

'nextpos': nextpos, 'next\_sentiment': next\_sentiment}

restaurant\_trial: ave Prec: 0.82, Rec: 0.74, F1: 0.78

laptop trial: ave Prec: 0.72, Rec: 0.62, F1: 0.66 (more Unicode issues there?)

**Results for subtask 2**

* 3/9, Features:
  + Overall sentence positive & negative sentiment score
  + Number of aspect terms
  + POS, word, iob, sentiment of word preceding & following aspect phrase
  + Word, pos & iob of head of aspect phrase
* 3/26, trying again now that some other issues have been worked out.
  + laptop 500 subset: 62.3%; goes up to 66% just using the better sentiment lexicon on the individual words, not even doing better sentence classification yet.
  + Simple split of laptop full, 65.8%; with dependency parse: 62.0%

**Adding baseline sentiment to laptop:**

Fold1: 0.60649 🡪 barely better

.691441441441 🡪 barely worse

acc: 0.662995594714 🡪 better

. 606126914661 🡪 better

.672839506173 🡪 better

**average acc: 0.64797886584 🡪 better**

**task4\_stask2.k\_fold('Laptop\_train\_v2.pkl')**

unexpected lexicon entry: ['type=weaksubj', 'len=1', 'word1=impassive', 'pos1=adj', 'stemmed1=n', 'polarity=negative', 'priorpolarity=weakneg']

next fold, split size: 2436/609

2436 training examples

defaultdict(<type 'int'>, {'positive': 207, 'neutral': 88, 'negative': 188, 'conflict': 10})

acc: 0.600405679513

next fold, split size: 2436/609

2436 training examples

defaultdict(<type 'int'>, {'positive': 206, 'neutral': 74, 'negative': 156, 'conflict': 8})

acc: 0.693693693694

next fold, split size: 2436/609

2436 training examples

defaultdict(<type 'int'>, {'positive': 182, 'neutral': 95, 'negative': 167, 'conflict': 10})

acc: 0.656387665198

next fold, split size: 2436/609

2436 training examples

defaultdict(<type 'int'>, {'positive': 182, 'neutral': 104, 'negative': 155, 'conflict': 16})

acc: 0.564551422319

next fold, split size: 2436/609

2436 training examples

defaultdict(<type 'int'>, {'positive': 199, 'neutral': 94, 'negative': 192, 'conflict': 1})

acc: 0.658436213992

**average acc: 0.634694934943**

**Adding baseline sentiment, improves!...but not all folds (this is restaurant)**

0.676900 for first fold

0.63671373 second (worse)

. 674897119342 (better)

0.64914586071 (slightly better)

0.607923497268 (slightly better)

**average acc: 0.649116159535 somewhat better**

**>>> task4\_stask2.k\_fold('Rest\_train\_v2.pkl')**

next fold, split size: 2432/609

2432 training examples

defaultdict(<type 'int'>, {'positive': 414, 'neutral': 114, 'negative': 137, 'conflict': 19})

acc: 0.666666666667 🡪 **NB, .646 🡪 DT .485!**

next fold, split size: 2433/608

2433 training examples

defaultdict(<type 'int'>, {'positive': 464, 'neutral': 138, 'negative': 166, 'conflict': 11})

acc: 0.637997432606 **🡪 NB .6187**

next fold, split size: 2433/608

2433 training examples

defaultdict(<type 'int'>, {'positive': 435, 'neutral': 116, 'negative': 157, 'conflict': 21})

acc: 0.657064471879

next fold, split size: 2433/608

2433 training examples

defaultdict(<type 'int'>, {'positive': 423, 'neutral': 119, 'negative': 201, 'conflict': 18})

acc: 0.645203679369

next fold, split size: 2433/608

2433 training examples

defaultdict(<type 'int'>, {'positive': 422, 'neutral': 146, 'negative': 142, 'conflict': 22})

acc: 0.607923497268

**average acc: 0.642971149558**

**Appendix**

**Graph traversal testing:**

body-5: 5, ['liked-2'] [('det', 'the-3'), ('nn', 'aluminum-4')]

ROOT-0: 0, [] [('root', 'liked-2')]

liked-2: 2, ['ROOT-0'] [('nsubj', 'I-1'), ('dobj', 'body-5')]

DISTANCES:

{'ROOT-0': {'I-1': 2, 'ROOT-0': 0, 'aluminum-4': 3, 'liked-2': 1, 'the-3': 3},

'body-5': {'I-1': 2, 'ROOT-0': 2, 'aluminum-4': 1, 'body-5': 0, 'liked-2': 1, 'the-3': 1},

'liked-2': {'I-1': 1, 'aluminum-4': 2, 'liked-2': 0, 'the-3': 2}}

ROOT-0: 0, [] [('root', 'beautiful-6')]

screen-4: 4, ['beautiful-6', 'Lightweight-1'] [('det', 'the-3')]

Lightweight-1: 1, ['beautiful-6'] [('conj\_and', 'screen-4')]

beautiful-6: 6, ['ROOT-0'] [('nsubj', 'Lightweight-1'), ('nsubj', 'screen-4'), ('cop', 'is-5')]

DISTs:

{'Lightweight-1': {'Lightweight-1': 0, 'beautiful-6': 1, 'is-5': 2, 'the-3': 2},

'ROOT-0': {'Lightweight-1': 2, 'ROOT-0': 0, 'beautiful-6': 1, 'is-5': 2, 'screen-4': 2, 'the-3': 3},

'beautiful-6': {'beautiful-6': 0, 'is-5': 1, 'the-3': 2},

'screen-4': {'Lightweight-1': 1, 'beautiful-6': 1, 'is-5': 2, 'screen-4':0 'the-3': 1}}

polarity dists:

the-3: 3, parents: ['screen-4'] sentiment: 2 / positive; []

screen-4: 4, parents: ['beautiful-6', 'Lightweight-1'] sentiment: 1 / positive

[('det', 'the-3')]

is-5: 5, parents: ['beautiful-6'] sentiment: 1 / positive; []

Lightweight-1: 1, parents: ['beautiful-6'] sentiment: 1 / positive

[('conj\_and', 'screen-4')]

ROOT-0: 0, parents: [] sentiment: 1 / positive

[('root', 'beautiful-6')]

beautiful-6: 6, parents: ['ROOT-0'] sentiment: 0 / positive

[('nsubj', 'Lightweight-1'), ('nsubj', 'screen-4'), ('cop', 'is-5')]

**Notes while preparing phase A submission**

3/25

Without dependency parsing, with sentiment lexicons and following feature set

'word': word, 'pos': pos, 'sentiment': sentiment, 'obj': objectivity,

'prevw': prevw, 'prevpos': prevpos, 'prevtag': prevtag, 'prev\_sentiment': prev\_sentiment, 'prev\_obj': prev\_obj,

'nextw': nextw, 'nextpos': nextpos, 'next\_sentiment': next\_sentiment, 'next\_obj': next\_obj}

**Maxent: on restaurant\_train\_v2,**

next fold, split size: 2432/609

# of vocab words: 4082 num examples: 609; num terms: 684; num guesses: 613

**Recall 0.73, Precision: 0.81, F1 0.76** **🡪 with dep parse: almost same, .7 Recall, same F**

next fold, split size: 2433/608;# of vocab words: 4047

num examples: 608;num terms: 779

num guesses: 655

Recall 0.68, Precision: 0.80, F1 0.73 **🡪 dep parse Recall 0.66, Precision: 0.80, F1 0.72**

next fold, split size: 2433/608

# of vocab words: 4093;num examples: 608

num terms: 729;num guesses: 617

Recall 0.68, Precision: 0.80, F1 0.74 🡪 dp Recall 0.66, Precision: 0.81, F1 0.73

next fold, split size: 2433/608

# of vocab words: 4058;num examples: 608

num terms: 761;num guesses: 633

Recall 0.69, Precision: 0.83, F1 0.75

next fold, split size: 2433/608

# of vocab words: 4070;num examples: 608

num terms: 732;num guesses: 603

Recall 0.69, Precision: 0.84, F1 0.76

**ave Prec: 0.82, Rec: 0.69, F1: 0.75**

**Naïve Bayes on rest\_train\_v2**

next fold, split size: 2432/609

num guesses: 1163

Recall 0.80, Precision: 0.47, F1 0.59

next fold, split size: 2433/608

num guesses: 1219

Recall 0.78, Precision: 0.50, F1 0.61

next fold, split size: 2433/608

num guesses: 1146

Recall 0.78, Precision: 0.50, F1 0.61

next fold, split size: 2433/608

num guesses: 1181

Recall 0.78, Precision: 0.51, F1 0.61

next fold, split size: 2433/608

num guesses: 1161

Recall 0.79, Precision: 0.50, F1 0.61

**ave Prec: 0.50, Rec: 0.79, F1: 0.61**

**Maxent on lap\_v2**

next fold, split size: 2436/609

# of vocab words: 3785

num examples: 609

num terms: 493

num guesses: 311

Recall 0.49, Precision: 0.78, F1 0.60

next fold, split size: 2436/609

# of vocab words: 3870

num examples: 609

num terms: 444

num guesses: 301

Recall 0.52, Precision: 0.77, F1 0.62

next fold, split size: 2436/609

# of vocab words: 3813

num examples: 609

num terms: 454

num guesses: 318

Recall 0.51, Precision: 0.73, F1 0.60

next fold, split size: 2436/609

# of vocab words: 3833

num examples: 609

num terms: 457

num guesses: 322

Recall 0.59, Precision: 0.84, F1 0.69

next fold, split size: 2436/609

# of vocab words: 3816

num examples: 609

num terms: 486

num guesses: 336

Recall 0.53, Precision: 0.76, F1 0.62

**ave Prec: 0.78, Rec: 0.53, F1: 0.63**

**DT: ave Prec: 0.61, Rec: 0.57, F1: 0.59**

**NB on lap\_v2**

Recall 0.70, Precision: 0.38, F1 0.50

Recall 0.68, Precision: 0.34, F1 0.45

Recall 0.69, Precision: 0.35, F1 0.46

Recall 0.76, Precision: 0.37, F1 0.49

Recall 0.72, Precision: 0.36, F1 0.48

ave Prec: 0.36, Rec: 0.71, F1: 0.48