More Smoothing, Tuning, and Evaluation

Nathan Schneider
(slides adapted from Henry Thompson, Alex Lascarides, Chris Dyer, Noah Smith, et al.)
ENLP | 21 September 2016

Review:

Naïve Bayes Classifier

$w_j \leftarrow [\mathbf{words}(x)]_j$

return

$\arg\max_{y'} \mathrm{p}(y') \times \Pi_j\, \mathrm{p}(w_j \mid y')$

$x \rightarrow$ *C* $\rightarrow y$

Naïve Bayes Learner

$X \rightarrow$

$Y \rightarrow$

$$\forall y,\ \mathrm{p}(y) \leftarrow \frac{\mathrm{count}(y)}{N}$$

$$\mathrm{p}(\ ) \leftarrow \frac{(\#\ \text{in } Y)}{(\#\ \text{docs in } X)}$$

$$\forall y,\ \forall w,\ \mathrm{p}(w \mid y) \leftarrow \frac{\mathrm{count}(w, y)}{\mathrm{count}(y)}$$

$$\mathrm{p}(\text{horrific} \mid\ ) \leftarrow \frac{(\#\ \text{docs with horrific})}{(\#\ \text{docs})}$$

L

$\rightarrow$ p

Smoothing p(w | y)

p(horrific | 👍) ← $\dfrac{\text{(\# 👍 docs with horrific)}}{\text{(\# 👍 docs)}}$

- What if we encounter the word distraught in a test document, but it has never been seen in training?
  - Can't estimate p(distraught | 👍) or p(distraught | 👎): numerator will be 0
  - Because the word probabilities are multiplied together for each document, the probability of the whole document will be 0

Smoothing p(w | y)

$$p(\text{horrific} \mid \text{👍}) \leftarrow \frac{(\#\ \text{👍 docs with horrific}) + 1}{(\#\ \text{👍 docs}) + V + 1}$$

$$p(\text{OOV} \mid \text{👍}) \leftarrow \frac{1}{(\#\ \text{👍 docs}) + V + 1}$$

V is the size of the vocabulary of the training corpus

- **Smoothing** techniques adjust probabilities to avoid **overfitting** to the training data
  - Above: **Laplace (add-1) smoothing**
  - OOV (out-of-vocabulary/unseen) words now have small probability, which decreases the model's confidence in the prediction without ignoring the other words
  - Probability of each seen word is reduced slightly to save probability mass for unseen words

Smoothing p(w | y)

$$p(\text{horrific} \mid \text{👍}) \leftarrow \frac{(\#\ \text{👍}\ \text{docs with horrific}) + 1}{(\#\ \text{👍}\ \text{docs}) + V + 1}$$

$$p(\text{OOV} \mid \text{👍}) \leftarrow \frac{1}{(\#\ \text{👍}\ \text{docs}) + V + 1}$$

V is the size of the vocabulary of the training corpus

- **Smoothing** techniques adjust probabilities to avoid **overfitting** to the training data
  - Above: **Laplace (add-1) smoothing**
  - OOV (out-of-vocabulary/unseen) words now have small probability, which decreases the model's confidence in the prediction without ignoring the other words
  - Probability of each seen word is reduced slightly to save probability mass for unseen words

New:

Smoothing p(w | y)

$$p(\text{horrific} \mid\ ) \leftarrow \frac{(\#\ \text{docs with horrific}) + 1}{(\#\ \text{docs}) + V + 1}$$

$$p(\text{OOV} \mid\ ) \leftarrow \frac{1}{(\#\ \text{docs}) + V + 1}$$

V is the size of the vocabulary of the training corpus

- **Laplace (add-1) smoothing**, above, uses a **pseudo-count** of 1, which is kind of arbitrary.
  - For some datasets, it's overkill—better to smooth less.
  - **Lidstone (add-α) smoothing**: **tune** the amount of smoothing on **development** data:

$$p(\text{horrific} \mid\ ) \leftarrow \frac{(\#\ \text{docs with horrific}) + \alpha}{(\#\ \text{docs}) + \alpha(V + 1)}$$

$$p(\text{OOV} \mid\ ) \leftarrow \frac{\alpha}{(\#\ \text{docs}) + \alpha(V+1)}$$

The Nature of Evaluation

- Scientific method rests on making and testing hypotheses.
- Evaluation is just another name for testing.

The Nature of Evaluation

- Scientific method rests on making and testing hypotheses.
- Evaluation is just another name for testing.
- Evaluation not just for public review:
  - It's how you manage internal development
  - And even how systems improve themselves (see ML courses).

What Hypotheses?

About existing linguistic objects:

- Is this text by Shakespeare or Marlowe?

About output of a language system:

- How well does this language model predict the data?
- How accurate is this segmenter/tagger/parser?
  - Is this segmenter/tagger/parser better than that one?

About human beings:

- How reliable is this person's annotation?
- To what extent do these two annotators agree? (IAA)

Gold Standard Evaluation

- In many cases we have a record of 'the truth':
  - The best human judgement as to what the correct segmentation/tag/parse/reading is,
    or what the right documents are in response to a query.
- Gold standards used both for training and for evaluation
- But testing must be done on unseen data (held-out test set; train/test split)

Don't ever train on data that you'll use in testing!!

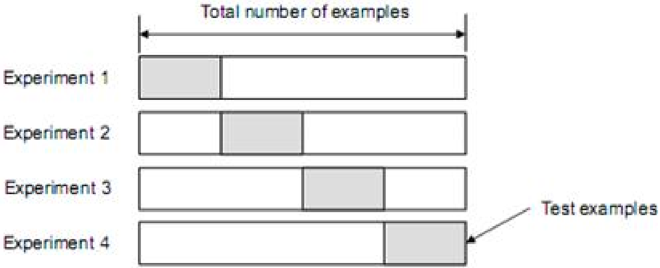
Tuning

- Often, in designing a system, you'll want to **tune** it by trying several configuration options and choosing the one that works best empirically.
  - E.g., Lidstone (add-λ) smoothing; choosing features for text classification.
- If you run several experiments on the test set, you risk **overfitting** it; i.e., the test set is no longer a reliable proxy for new data.
- One solution is to hold out a second set for tuning, called a **development** ("dev") set. Save the test set for the very end.

Cross-validation

What if my dataset is **too small** to have a nice train/test or train/dev/test split?

- *k*-**fold cross-validation**: partition the data into *k* pieces and treat them as mini held-out sets. Each **fold** is an experiment with a different held-out set, using the rest of the data for training:



Measuring a Model's Performance

Accuracy

Proportion model gets right:

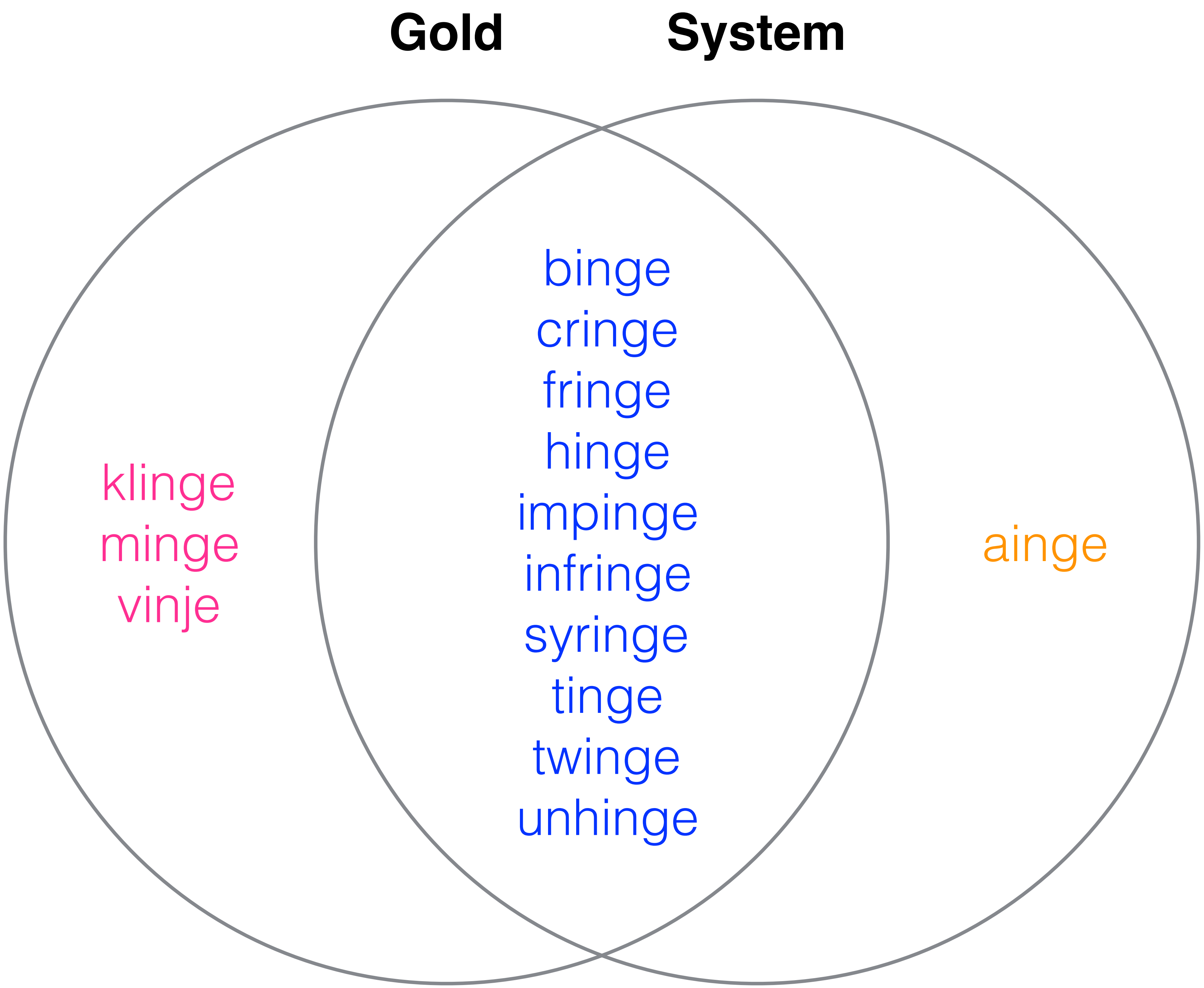
$$\frac{|\text{right}|}{|\text{test-set}|} \times 100$$

E.g., POS tagging (state of the art $\approx 96\%$).

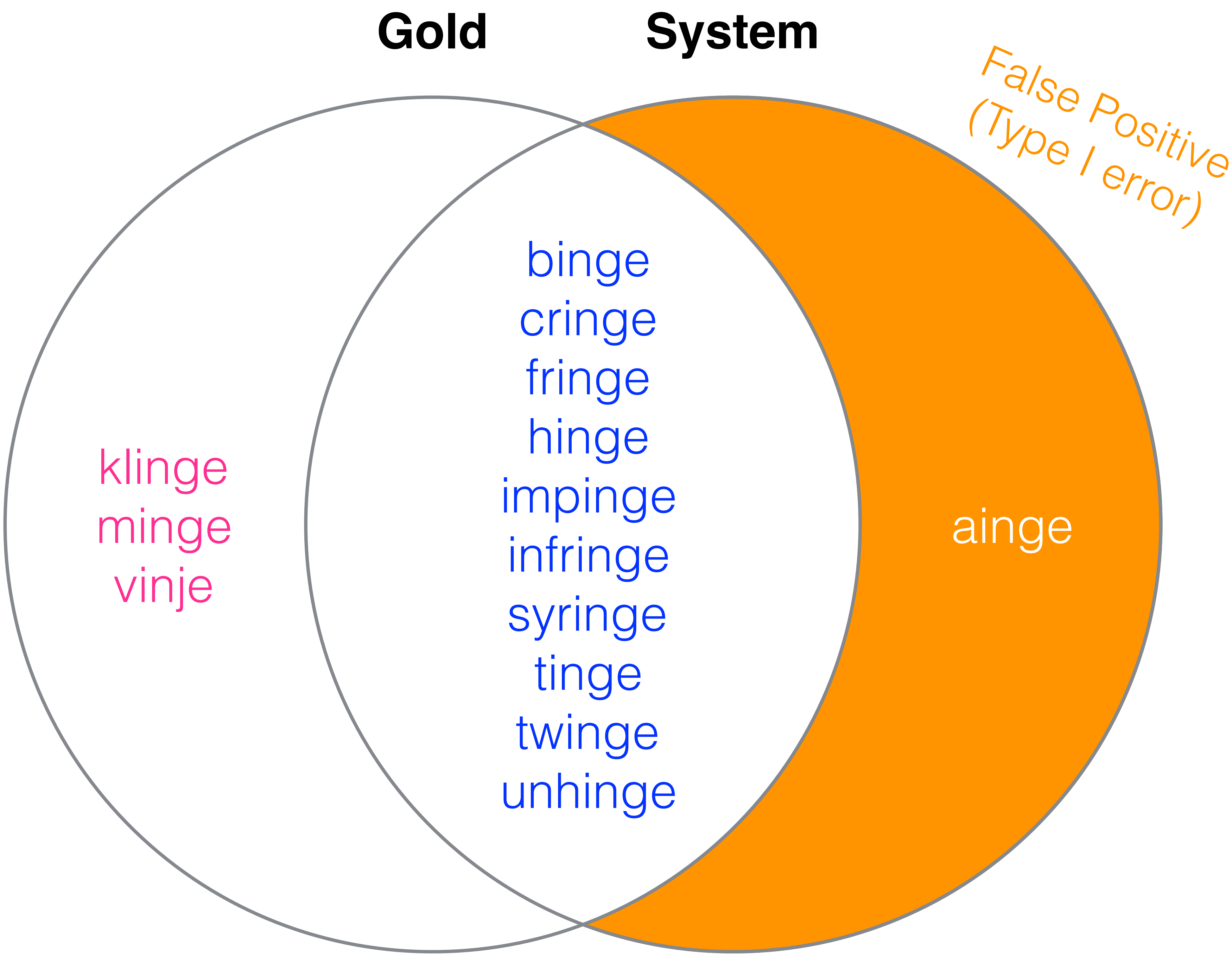
How should we evaluate your rhyming script?

- Suppose somebody creates a gold standard of <input_word, {rhyming_words}> pairs.
- Multiple desired outputs; system's outputs could overlap only partially. How to evaluate?
  - Accuracy over all words in the dictionary?

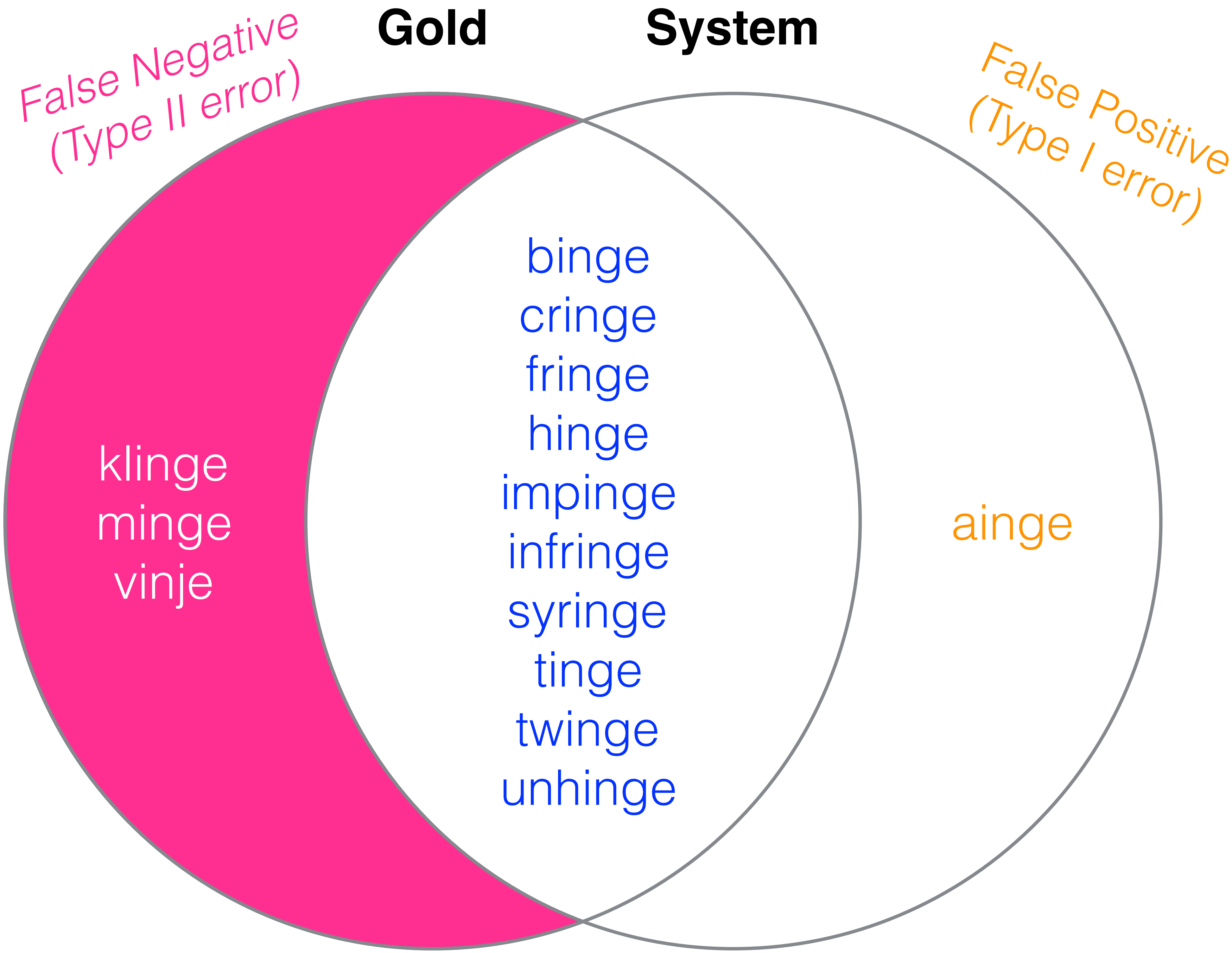
Rhymes for “hinge”



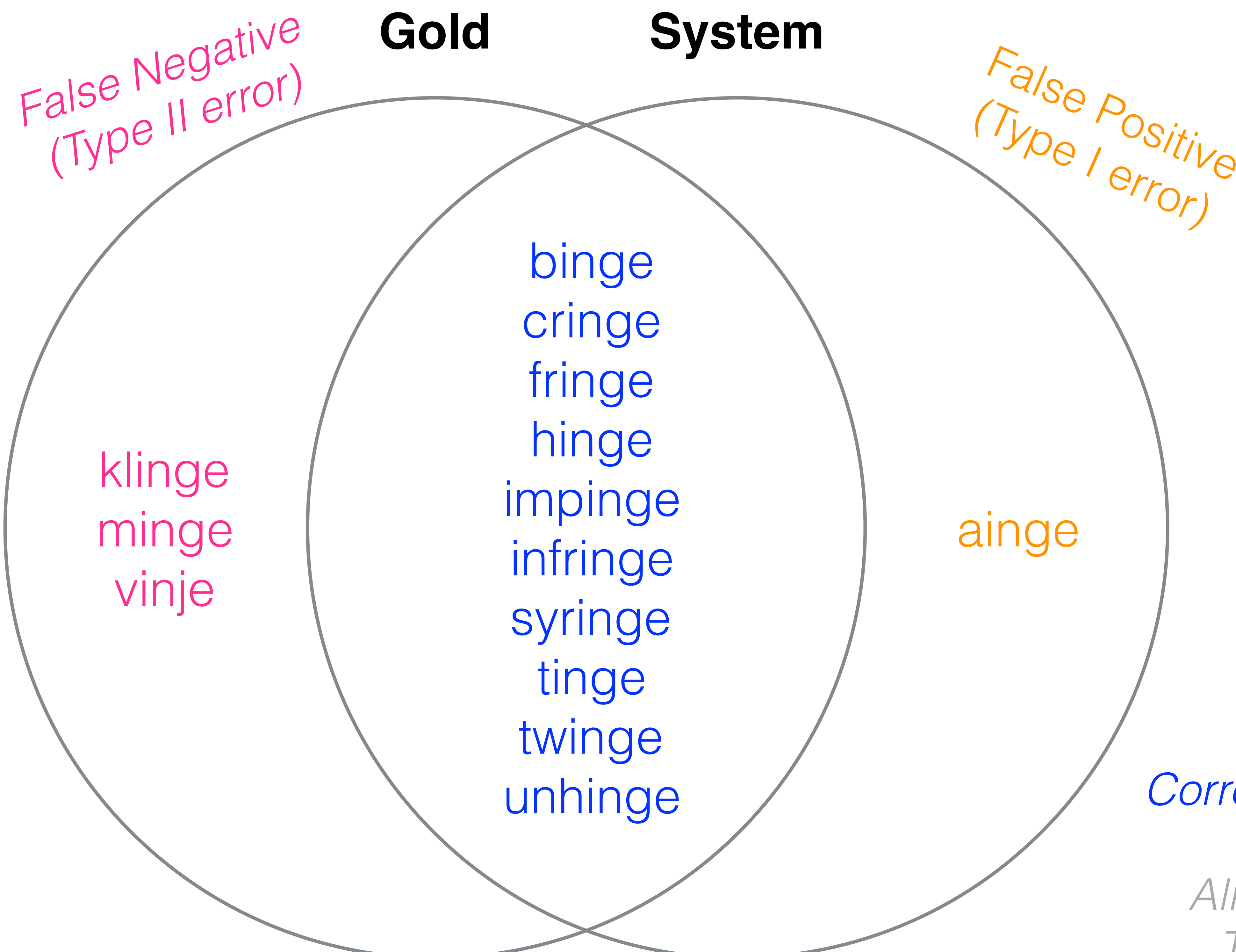
Rhymes for “hinge”



Rhymes for “hinge”



Rhymes for “hinge”

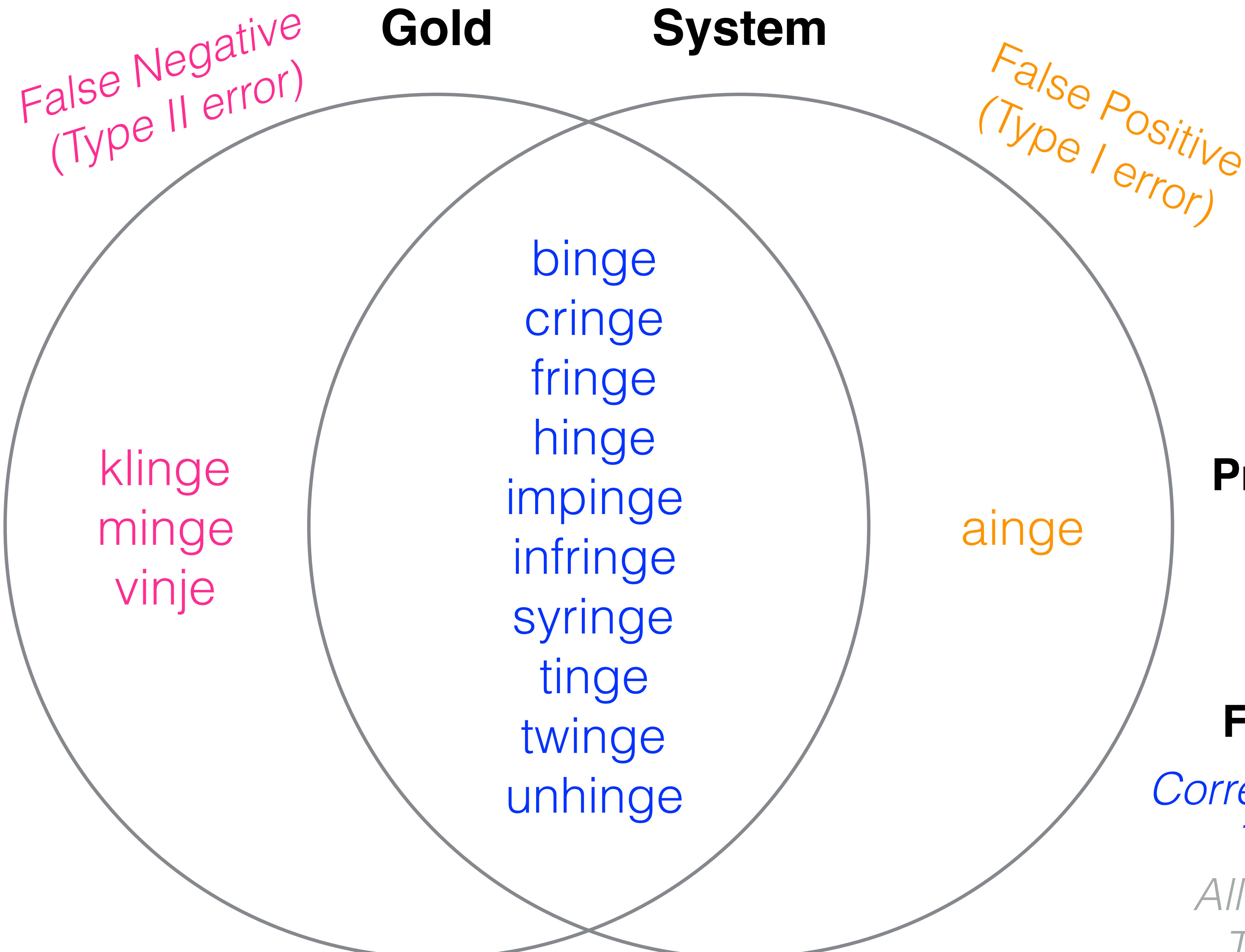


	Sys=Y	Sys=N
Gold=Y	10	3
Gold=N	1	(large)

Correctly predicted = True Positive

All other words = True Negative

Precision & Recall



	Sys=Y	Sys=N
Gold=Y	10	3
Gold=N	1	(large)

Precision = TP/(TP+FP)
= 10/11 = 91%

Recall = TP/(TP+FN)
= 10/13 = 77%

F_1 = 2·P·R/(P+R) = 83%

Correctly predicted =
True Positive

All other words =
True Negative

Measuring a Model's Performance

Precision, Recall, F-score

- For isolating performance on a particular label in multi-label tasks, or
- For chunking, phrase structure parsing, or anything where word-by-word accuracy isn't appropriate.
- F_1-score: Harmonic mean of precision (proportion of model's answers that are right) and recall (proportion of test data that model gets right).
- E.g., for the POS tag `NN`:

$$
\begin{aligned}
P &= \frac{|\text{tokens correctly tagged } \texttt{NN}|}{|\text{all tokens automatically tagged } \texttt{NN}|} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \\
R &= \frac{|\text{tokens correctly tagged } \texttt{NN}|}{|\text{all tokens gold-tagged } \texttt{NN}|} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \\
F_1 &= \frac{P \cdot R}{P+R}
\end{aligned}
$$

Upper Bounds, Lower Bounds?

Suppose your POS tagger has 95% accuracy? Is that good? Bad??

Upper Bound: Turing Test

When using a human Gold Standard, check the agreement of humans against that standard.

Lower Bound: Performance of a 'simpler' model (**baseline**)

- Model always picks most frequent class (**majority baseline**).
- Model assigns a class randomly according to:
  1. Even probability distribution; or
  2. Probability distribution that matches the observed one.

Suitable upper and lower bounds depend on the task.

Measurements: What's Significant?

- We'll be measuring things, and comparing measurements.
- What and how we measure depends on the task.
- But all have one issue in common:

  Are the differences we find significant?

- In other words, should we interpret the differences as down to pure chance? Or is something more going on?
- Is our model significantly better than the baseline model?
  Is it significantly worse than the upper bound?

Example: Tossing a Coin

- I tossed a coin 40 times; it came up heads 17 times.
- Expected value of fair coin is 20. So we're comparing 17 and 20.
- If this difference is *significant*, then it's (probably) not a fair coin. If not, it (probably) is.

Which Significance Test?

- Paremetric when the underlying distribution is normal.
  - t-test, z-test,...
  - You don't need to know the mathematical formulae;
    available in statistical libraries!
- Non-Parametric otherwise.
  - Usually do need non-parametric tests:
    remember Zipf's Law!
  - Can use McNemar's test or variants of it.

See Smith (2011, Appendix B) for a detailed discussion of significance testing methods for NLP.

Error Analysis

- Summary scores are important, but don't always tell the full picture!
- Once you've built your system, it's always a good idea to dig into its output to identify patterns.
  - Quantitative *and* qualitative (look at some examples!)
  - You may find bugs (e.g., predictions are always wrong for words with accented characters)

Confusion Matrices

		Estimated Emotion							
		Anger	Boredom	Disgust	Fear	Happiness	Sadness	Neutral	*Emotion Recog. Rate*
True Emotion	Anger	**19**	0	2	0	**3**	0	0	79.2%
	Boredom	1	**8**	1	1	0	1	**7**	42.1%
	Disgust	0	1	**6**	0	1	0	3	54.5%
	Fear	1	3	2	**7**	2	0	1	43.8%
	Happiness	**3**	0	3	2	**5**	0	2	33.3%
	Sadness	0	0	0	0	0	**14**	0	100.0%
	Neutral	0	**5**	1	0	0	0	**13**	68.4%
	HMM Recog. Rate	79.2%	47.1%	40.0%	70.0%	45.5%	93.3%	50.0%	

Tasks where there is > 1 right answer

Example: A Paraphrasing Task

- Estimate that *John enjoyed the book* means *John enjoyed reading the book*.
- Lots of closely related words to *read* are good too:
  skim through, go through, peruse, etc.