Implementing Sentiment Analysis with a Rule-based Approach

Vitthal Srinivasan
CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Focus on using existing sentiment analysis tools

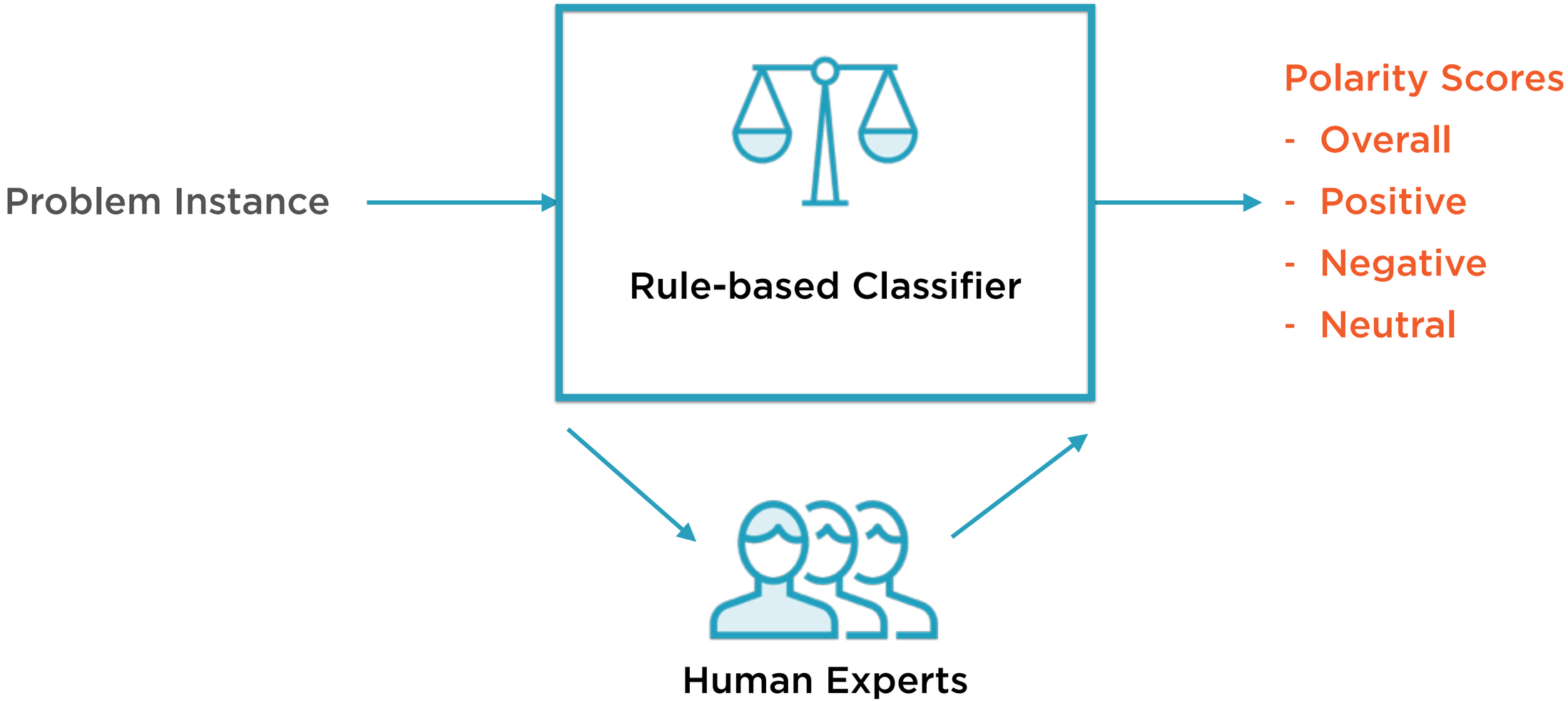
Analyse a dataset of 10,000+ movie reviews

- Using VADER
- Using Sentiwordnet

Appreciate the need for ML-based classifiers

Introducing VADER

VADER Is a Rule-based Binary Classifier



VADER

Valence Aware Dictionary for sEntiment Reasoning

Built into Python nltk

Created at Georgia Tech

Incredibly convenient to use

Both an algorithm and a dataset

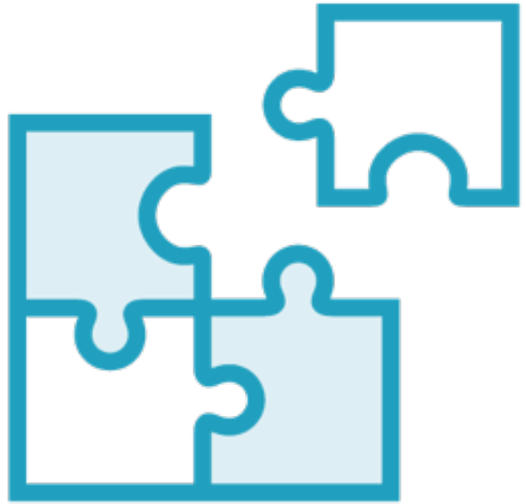
Limitations of a Simplistic Approach

Intensity

Valency, boosters, punctation, capitalisation

Reversal

Negation, contrasting conjunctions and adverbs



Context

Different meanings in different contexts

Both Algorithm and Dataset

Algorithm

Takes in text, calculates polarity and valence using rules

Dataset

Lexicon with valence scores for words, idioms, emoticons

Demo

Set up VADER for use in Python

Understand VADER support for

- Emoticons
- Idioms

Punctuation

Negation

Emphasis

Contrast

VADER as Algorithm

Words

Emoticons

Idioms

Introducing Sentiwordnet

Sentiwordnet

Sentiment lexicon with polarity information

Built into Python nltk

Very convenient to use

Extends wordnet

WordNet

“WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations.”
http://wordnet.princeton.edu/

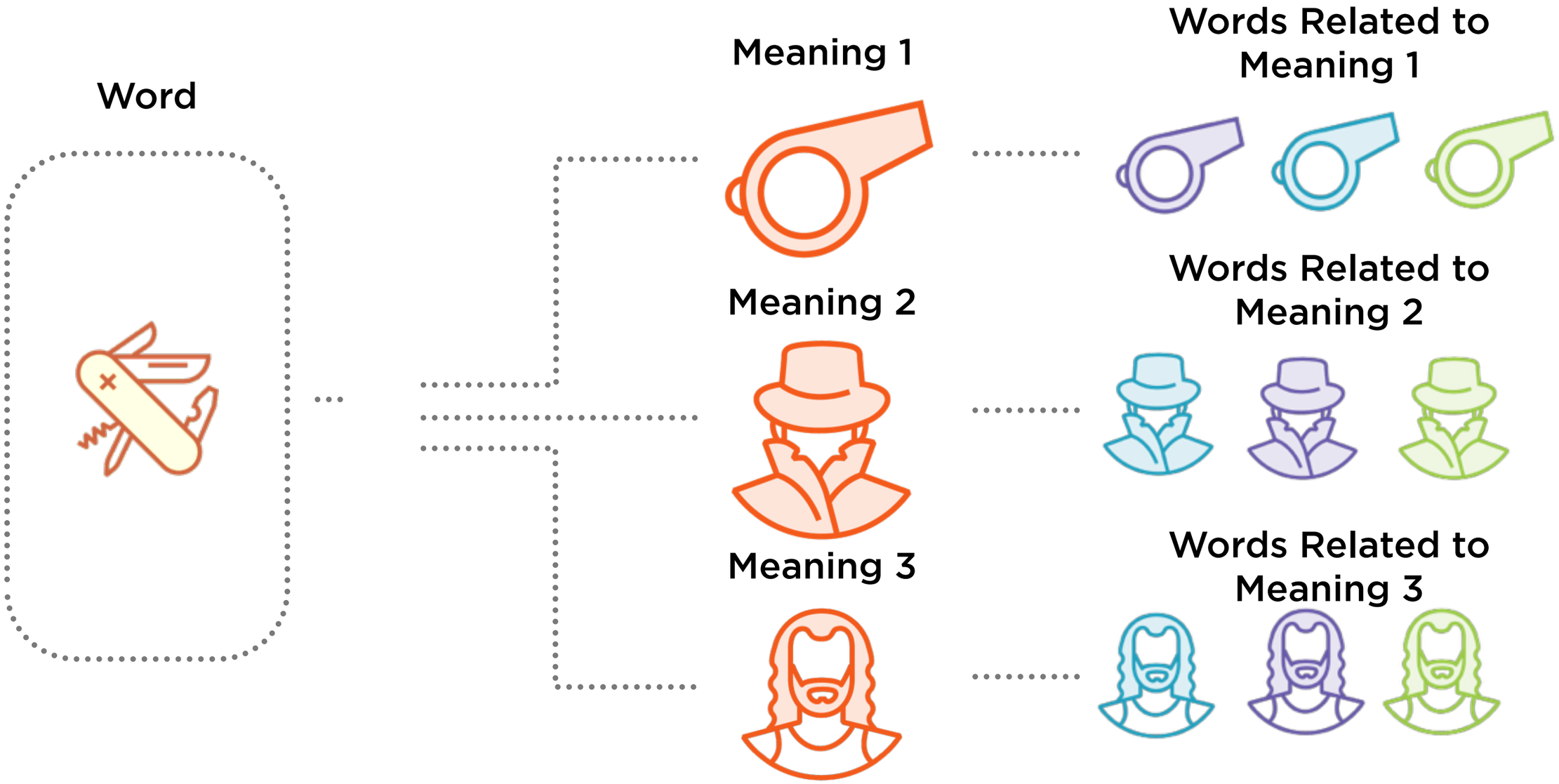
One Word, Many Meanings

“Love me, love my dog”

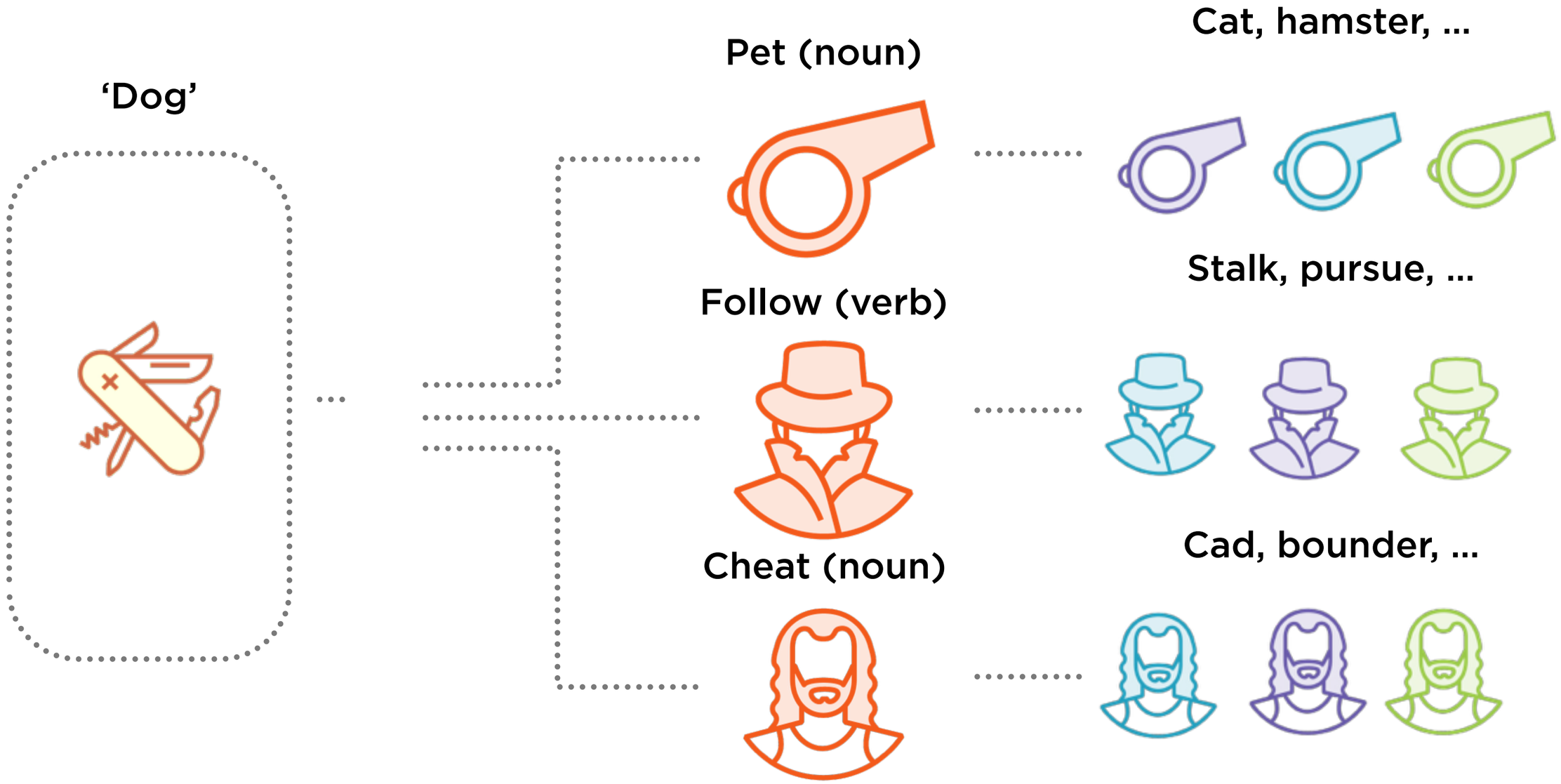
“Quit dogging my steps”

“He's a playa dog”

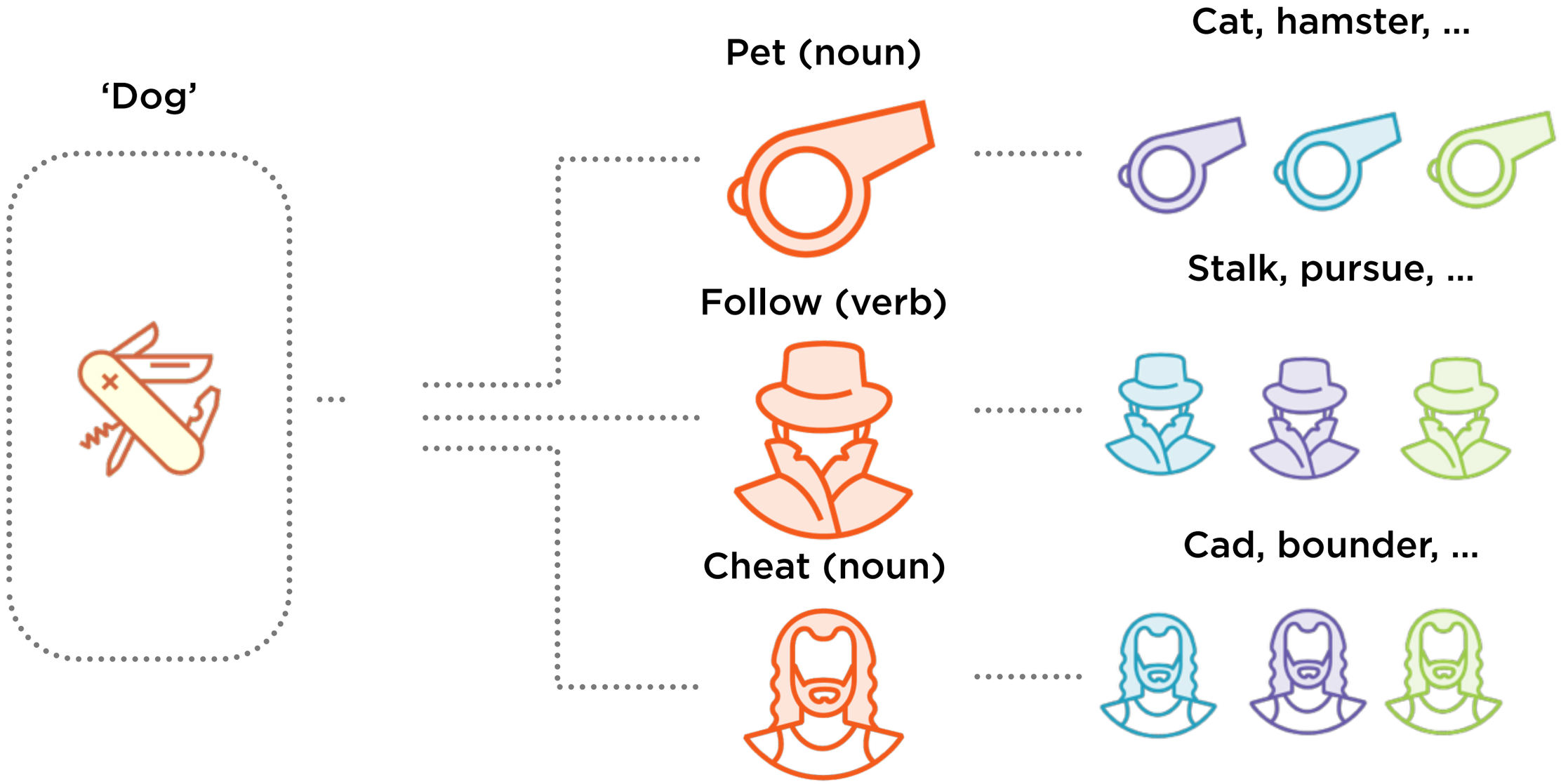
Words, Lemmas, Synsets



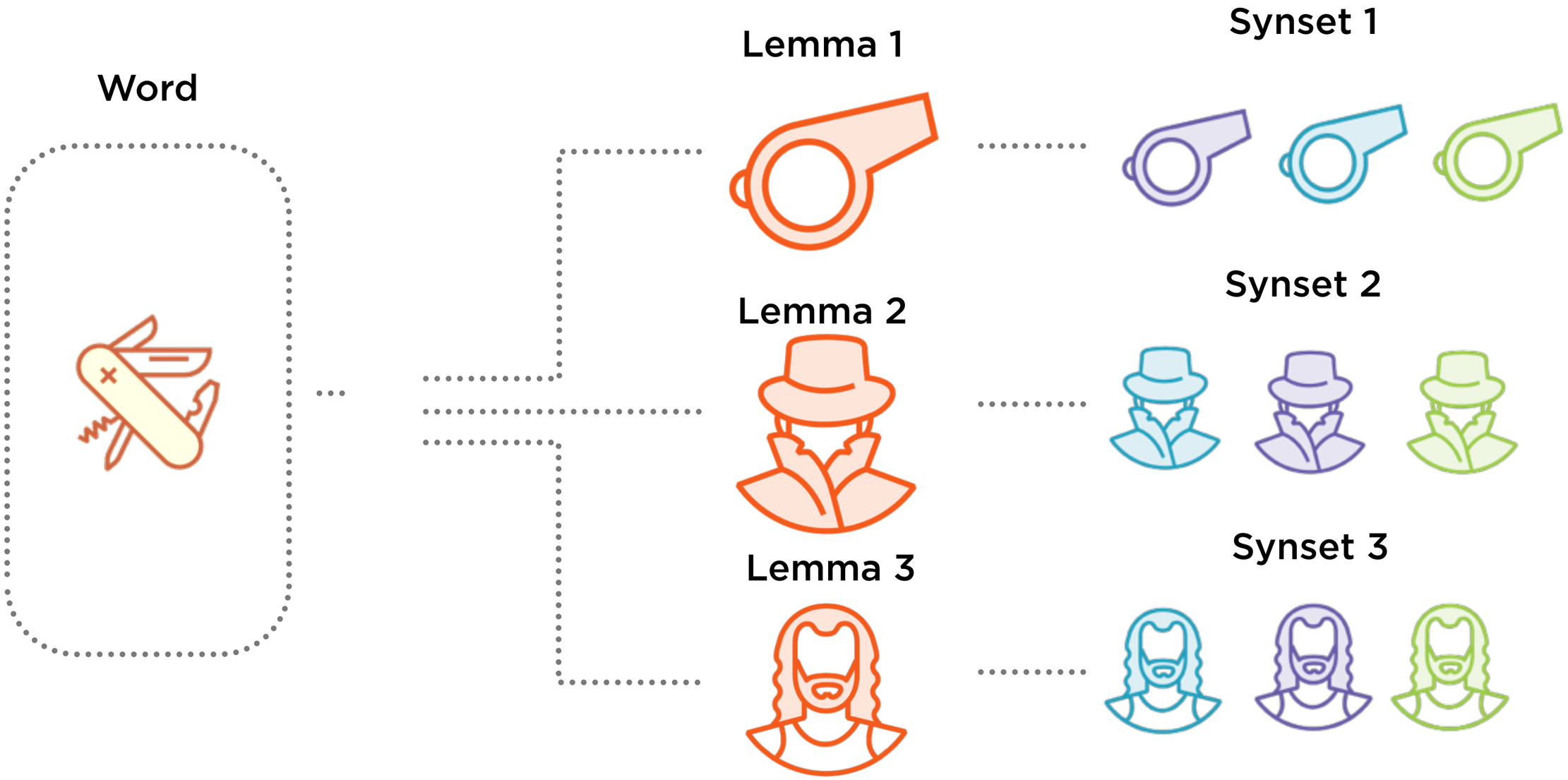
Words, Lemmas, Synsets



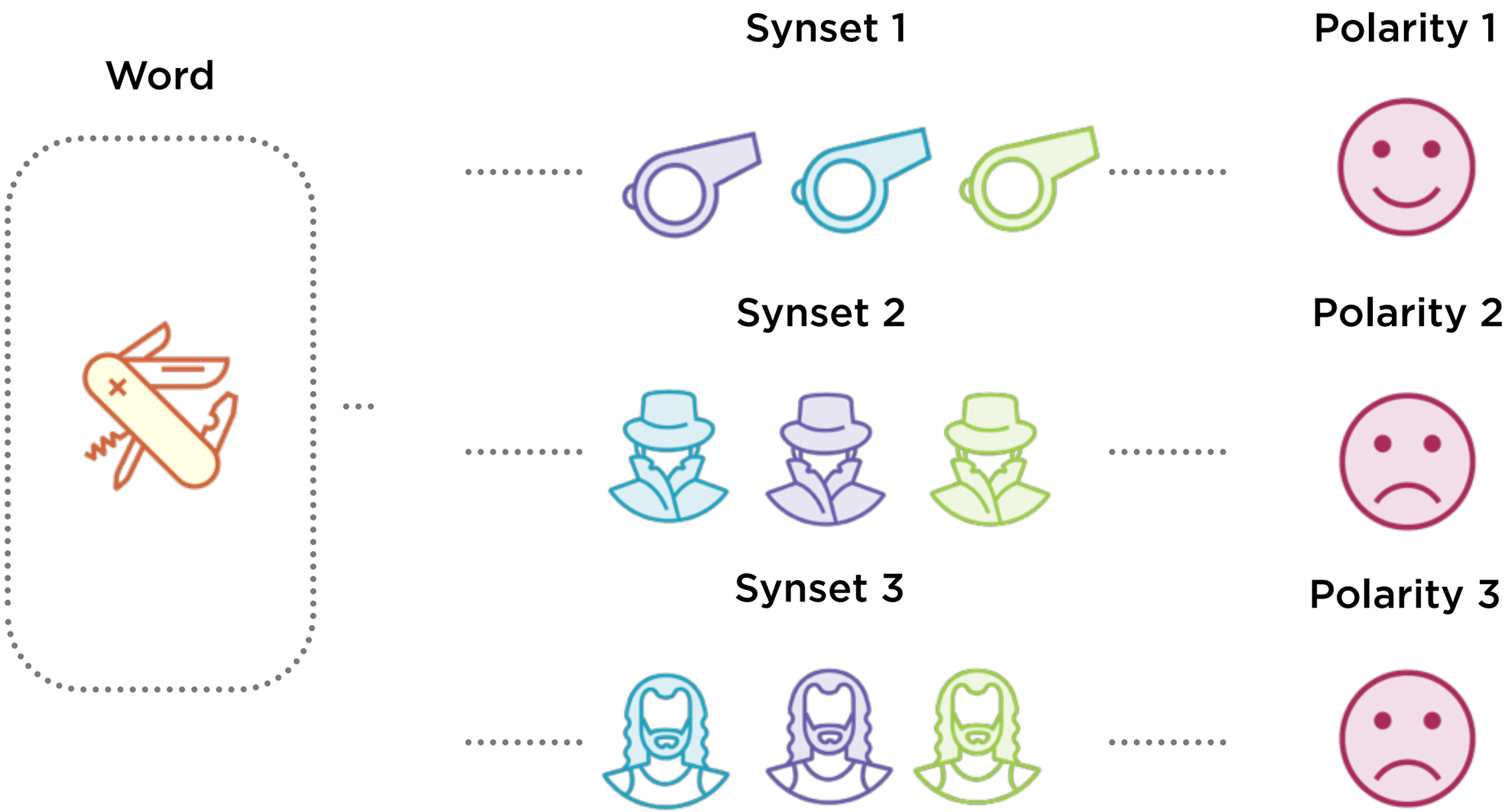
Words, Lemmas, Synsets



Words, Lemmas, Synsets



Sentiwordnet on Wordnet



One Word, Many Lemmas

“Love me, love my dog”

“Quit dogging my steps”

“He's a playa dog”

Demo

Use Sentiwordnet to look up polarities

Implementing A Rule-based Approach Using VADER

Demo

Download a corpus of pre-classified movie reviews

Apply VADER to this dataset

Calculate percentage accuracy

VADER on Cornell's Movie Review Data

Download Corpus

Cornell dataset of 10,000+ pre-classified movie reviews

Classify with VADER

Positive or negative? Use VADER's compound score to decide

Measure Accuracy

Calculate percentage accuracy

```
import nltk
from nltk.sentiment import vader
```

Import nltk

We can now invoke VADER from our code

```
sia = vader.SentimentIntensityAnalyzer()
def vaderSentiment(review):
  return sia.polarity_scores(review)['compound']
```

A Simple Function Invoking VADER

Rely entirely on the compound score

```
positiveReviewsFileName =
“/Users/vitthalsrinivasan/rt-polaritydata/rt-polaritydata/rt-polarity.pos”

with open(positiveReviewsFileName,'r') as f:
    positiveReviews = f.readlines()
```

Read in the Reviews

All positive reviews are now in a list named positiveReviews

```
negativeReviewsFileName =
"/Users/vitthalsrinivasan/rt-polaritydata/rt-polaritydata/rt-polarity.neg"

with open(negativeReviewsFileName,'r') as f:
    negativeReviews = f.readlines()
```

Read in the Reviews

Do the same with the negative reviews

```
[vaderSentiment(oneNegativeReview) for oneNegativeReview in negativeReviews]
```

Apply VADER to the Reviews

Python syntax for applying a function to each element of a list

```
[vaderSentiment(onePositiveReview) for onePositiveReview in positiveReviews]
```

Apply VADER to the Reviews

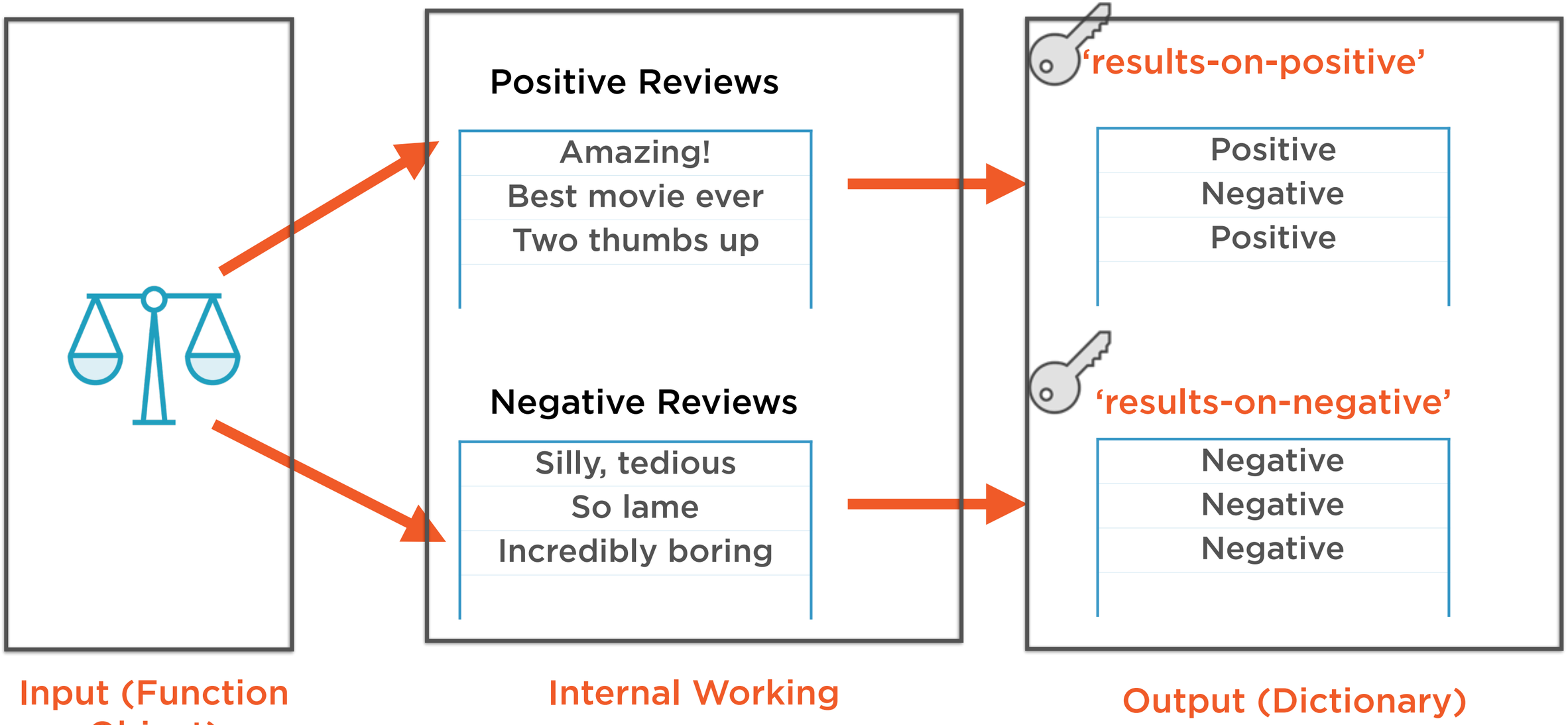
Python syntax for applying a function to each element of a list

```
def getReviewSentiments(sentimentCalculator):
  negReviewResult =
  [sentimentCalculator(oneNegativeReview) for oneNegativeReview in negativeReviews]
  posReviewResult =
  [sentimentCalculator(onePositiveReview) for onePositiveReview in positiveReviews]
  return {'results-on-positive':posReviewResult, 'results-on-negative':negReviewResult}
```

Code Reuse with **getReviewSentiments**

Create a function that takes in a function object and applies to all reviews

Code Reuse with **getReviewSentiments**



% of positive reviews that VADER classified correctly

```
pctTruePositive =
float(sum(x > 0 for x in positiveReviewsResult))/len(positiveReviewsResult)
```

Calculate Accuracy on Positive Reviews

Percentage of positive reviews that VADER classified as positive (correctly)

Numerator = number of positive reviews where VADER assigned positive polarity

```
pctTruePositive =
float(sum(x > 0 for x in positiveReviewsResult))/len(positiveReviewsResult)
```

Calculate Accuracy on Positive Reviews

Percentage of positive reviews that VADER classified as positive (correctly)

Denominator = total number of positive reviews

```
pctTruePositive =
float(sum(x > 0 for x in positiveReviewsResult))/len(positiveReviewsResult)
```

Calculate Accuracy on Positive Reviews

Percentage of positive reviews that VADER classified as positive (correctly)

```
pctTrueNegative =
float(sum(x < 0 for x in negativeReviewsResult))/len(negativeReviewsResult)
```

Calculate Accuracy on Negative Reviews

Percentage of negative reviews that VADER classified as negative (correctly)

```
totalAccurate =
float(sum(x > 0 for x in positiveReviewsResult))
+ float(sum(x < 0 for x in negativeReviewsResult))
total = len(positiveReviewsResult) + len(negativeReviewsResult)
print "Overall accuracy = " + "%.2f" % (totalAccurate*100/total) + "%"
```

Calculate Overall Accuracy

Percentage of negative reviews that VADER classified as negative (correctly)

```
def runDiagnostics(reviewResult):
  positiveReviewsResult = reviewResult['results-on-positive']
  negativeReviewsResult = reviewResult['results-on-negative']
  pctTruePositive = float(sum(x > 0 for x in positiveReviewsResult))/len(positiveReviewsResult)
  pctTrueNegative = float(sum(x < 0 for x in negativeReviewsResult))/len(negativeReviewsResult)
  totalAccurate = float(sum(x > 0 for x in positiveReviewsResult)) +
  float(sum(x < 0 for x in negativeReviewsResult))
  total = len(positiveReviewsResult) + len(negativeReviewsResult)
  print "Accuracy on positive reviews = " +"%.2f" % (pctTruePositive*100) + "%"
  print "Accuracy on negative reviews = " +"%.2f" % (pctTrueNegative*100) + "%"
  print "Overall accuracy = " + "%.2f" % (totalAccurate*100/total) + "%"
```

Allow for Code Reuse

Wrap calculation in a nice function available for reuse

Implementing A Rule-based Approach Using Sentiwordnet

Demo

Build a binary classifier using Sentiwordnet

Run this classifier on our corpus of reviews

Compare accuracy of VADER- and Sentiwordnet-based classifiers

Coming Up: ML-based Binary Classifier

Corpus

Classification Algorithm

ML-based Classifier

Summary

VADER is a powerful, easy-to-use rule-based classifier

VADER performs reasonably - but not spectacularly - on large datasets

Sentiwordnet can also be used to build powerful classifiers

Rule-based approaches do not exploit patterns in specific datasets