Natural Language Processing

Tynan Fitzpatrick

SBU ID#105735755

CSE 352 — Artificial Intelligence
Prof. Anita Wasiewska
October 23, 2008

Works Cited

- Bautin, Mikhail, Steven Skiena, et al. "Advances in News and Blog Analysis with Lydia." Stony Brook University. 20 Oct 2008 ⟨ http://cs.sunysb.edu⟩ .
- Carnie, Andrew. *Syntax: A Generative Introduction*. 2nd Edition. Malden, MA: Blackwell Publishing, 2007.
- Rabiner, Lawrence R. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." University of California, Santa Barbara. 20 Oct 2008 ⟨ http://www.ece.ucsb.edu⟩ .
- Sipser, Michael. *Introduction to the Theory of Computation*. 2nd Edition. Boston: Thompson, 2006.
- Skiena, Steve. *The Algorithm Design Manual*. 2nd Edition. London: Springer, 2008.
- "The Trouble with NLP." SpecGram. 20 Oct 2008 ⟨ http://specgram.com ⟩.

Outline

Warning!

- This presentation contains a good deal of technical content.

Warning!

- This presentation contains a good deal of technical content.
- Some of it may be from courses you have taken, but other material comes from linguistics and other areas.

Warning!

- This presentation contains a good deal of technical content.
- Some of it may be from courses you have taken, but other material comes from linguistics and other areas.
- I have tried to make as few assumptions as possible about your prior knowledge.

Warning!

- This presentation contains a good deal of technical content.
- Some of it may be from courses you have taken, but other material comes from linguistics and other areas.
- I have tried to make as few assumptions as possible about your prior knowledge.
- Of course, questions are welcome at any time.

How Do We Think About Language?

- Language is a remarkable thing.

How Do We Think About Language?

- Language is a remarkable thing.
- We are all able to produce an infinitude of sentences with barely any conscious thought.

How Do We Think About Language?

- Language is a remarkable thing.
- We are all able to produce an infinitude of sentences with barely any conscious thought.
- Despite the fact that no one has infinite brain capacity, we are all able to understand each other.

How Do We Think About Language?

- Language is a remarkable thing.
- We are all able to produce an infinitude of sentences with barely any conscious thought.
- Despite the fact that no one has infinite brain capacity, we are all able to understand each other.
- How?

Generative Grammar

- In Noam Chomsky's *Syntactic Structures* (1957), he claims that we cannot use word chains, databases or other similar devices.

Generative Grammar

- In Noam Chomsky's *Syntactic Structures* (1957), he claims that we cannot use word chains, databases or other similar devices.
- Instead, we need to have a set of rules (a grammar) that can generate any sentence that speakers can produce.

Generative Grammar

- In Noam Chomsky's *Syntactic Structures* (1957), he claims that we cannot use word chains, databases or other similar devices.
- Instead, we need to have a set of rules (a grammar) that can generate any sentence that speakers can produce.
- This lead to the creation of what we computer scientists call a context-free grammar.

What is a Context-Free Grammar?

- A context-free grammar (or CFG) is a series of rules which define how symbols that define internal structure can be replaced with symbols from the language that define the external structure. (Sipser 99)

What is a Context-Free Grammar?

- A context-free grammar (or CFG) is a series of rules which define how symbols that define internal structure can be replaced with symbols from the language that define the external structure. (Sipser 99)
- The primary advantage of CFGs over regular expressions is that they are able to express recursion.

What is a Context-Free Grammar?

- A context-free grammar (or CFG) is a series of rules which define how symbols that define internal structure can be replaced with symbols from the language that define the external structure. (Sipser 99)
- The primary advantage of CFGs over regular expressions is that they are able to express recursion.
- That's a rather weighty defintion — here's an example:

$$P \rightarrow (P)|(P)(P)|\epsilon$$

What is a Context-Free Grammar?

- A context-free grammar (or CFG) is a series of rules which define how symbols that define internal structure can be replaced with symbols from the language that define the external structure. (Sipser 99)
- The primary advantage of CFGs over regular expressions is that they are able to express recursion.
- That's a rather weighty defintion — here's an example:

$$P \rightarrow (P)|(P)(P)|\epsilon$$

- This is the grammar that generates all matched lists of parentheses.

Languages and Context-Free Grammar

- Chomsky proposed that language can also be specified by a CFG.

Languages and Context-Free Grammar

- Chomsky proposed that language can also be specified by a CFG.
- This is the modern representation of the rules that he gave for English:

$$\begin{aligned} XP &\rightarrow (SP)X' \\ X' &\rightarrow X'(AP)|(AP)X' \\ X' &\rightarrow X(CP) \end{aligned}$$

(Carnie 175)

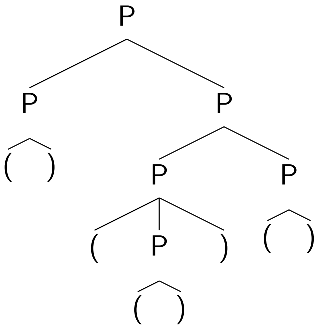
Languages and Context-Free Grammar

- Chomsky proposed that language can also be specified by a CFG.
- This is the modern representation of the rules that he gave for English:

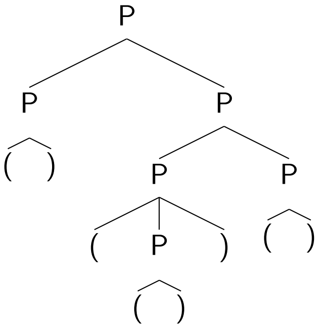
$$\begin{array}{rcl} XP & \rightarrow & (SP)X' \\ X' & \rightarrow & X'(AP)|(AP)X' \\ X' & \rightarrow & X(CP) \end{array}$$

  (Carnie 175)
- Let's explore the meaning of this grammar via means of analogy.

Derivation Tree for ()(())()

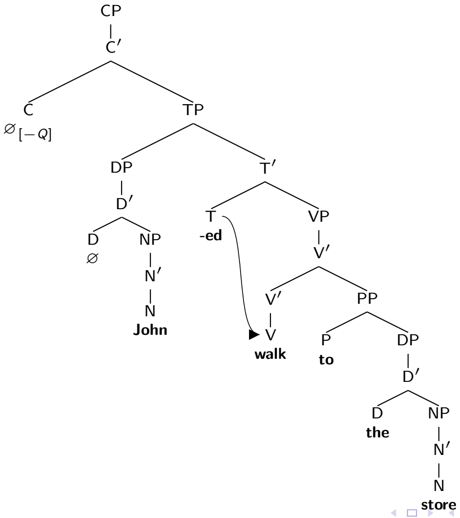
- 

Derivation Tree for ()(())()

- 

- This is just one of several ways of deriving this statement — hence this is an ambiguous grammar.

Derivation Tree for an English Sentence

- 

Implications of English Syntax

- Chomskyian grammar allows us to gain a good computational foothold on the English language.

Implications of English Syntax

- Chomskyian grammar allows us to gain a good computational foothold on the English language.
- For simple grammars we can use a stack-based implementation — for example, the parethesis-matching grammar above can be recognized by pushing every left parenthesis on the stack, and popping off a parenthesis when a right parenthesis is read.

Implications of English Syntax

- Chomskyian grammar allows us to gain a good computational foothold on the English language.
- For simple grammars we can use a stack-based implementation — for example, the parethesis-matching grammar above can be recognized by pushing every left parenthesis on the stack, and popping off a parenthesis when a right parenthesis is read.
- However, this strategy leads to nondeterministic behavior in general. We will need a more powerful computational tool. (Sipser 115)

Dynamic Programming and Context-Free Grammars

- Dynamic programming is a means of turning a recurrence with a polynomial number of possible arguments that takes exponential computation time into a polynomial algorithm.

Dynamic Programming and Context-Free Grammars

- Dynamic programming is a means of turning a recurrence with a polynomial number of possible arguments that takes exponential computation time into a polynomial algorithm.
- We do this by *memoizing* the results of the recurrence, so that when it is called again with the same arguments, we can just do a constant-time table lookup instead of repeating a lengthy computation.

Dynamic Programming and Context-Free Grammars

- Dynamic programming is a means of turning a recurrence with a polynomial number of possible arguments that takes exponential computation time into a polynomial algorithm.
- We do this by *memoizing* the results of the recurrence, so that when it is called again with the same arguments, we can just do a constant-time table lookup instead of repeating a lengthy computation.
- But what should our recurrence be? Although CFGs exhibit recursion, they don't immediately lend themselves to an intuitive recurrence.

Chomsky Normal Form

- To create this recurrence, we'll need to convert our grammar to something called the Chomsky Normal Form, which is the form where all rules are either of the form $X \rightarrow YZ$ or $X \rightarrow x$.

Chomsky Normal Form

- To create this recurrence, we'll need to convert our grammar to something called the Chomsky Normal Form, which is the form where all rules are either of the form $X \to YZ$ or $X \to x$.
- Any grammar can be converted to an equivalent Chomsky Normal Form by an entirely mechanical but somewhat tedious process. (Sipser 107)

Chomsky Normal Form

- To create this recurrence, we'll need to convert our grammar to something called the Chomsky Normal Form, which is the form where all rules are either of the form $X \longrightarrow YZ$ or $X \longrightarrow x$.
- Any grammar can be converted to an equivalent Chomsky Normal Form by an entirely mechanical but somewhat tedious process. (Sipser 107)
- Fortunately, natural language is already in a Chomsky Normal Form.

Creating A Recurrence

- Therefore, a couple of rules for parsing spring to mind:

Creating A Recurrence

- Therefore, a couple of rules for parsing spring to mind:
- If we try to parse a string using the rule $X \rightarrow x$, then that string can only contain the single character x.

Creating A Recurrence

- Therefore, a couple of rules for parsing spring to mind:
- If we try to parse a string using the rule $X \rightarrow x$, then that string can only contain the single character x.
- If we try to parse a string using the rule $X \rightarrow YZ$ and the string is in the grammar $((M(1, n, X)))$, then it must be the case that $\exists i (1 \leq i < n) \wedge (M(1, i, Y) \wedge M(i + 1, n, Z))$.

The Algorithm

- With these two rules, we can now create our dynamic programming array:

$$M[i,j,X] = \bigvee_{(X \to YZ) \in G} \left(\bigvee_{k=i}^{j} M[i,k,Y] \cdot M[k+1,j,Z] \right)$$

(Skiena 299)

The Algorithm

- With these two rules, we can now create our dynamic programming array:

$$M[i,j,X] = \bigvee_{(X \rightarrow YZ) \in G} \left(\bigvee_{k=i}^{j} M[i,k,Y] \cdot M[k+1,j,Z] \right)$$

(Skiena 299)

- Since it takes linear time to evaluate each cell in the array, and there are n^2 cells, evaluating $M(1,n,S)$, where S is the start symbol of the grammar, takes $\mathcal{O}(n^3)$ time.

Problems with Syntax Parsing

- With these constructs that we've seen, why is it that natural language can't be compiled like a programming language? After all, it would be difficult to program if a compiler performed as poorly as Microsoft Word's grammar checker.

Problems with Syntax Parsing

- With these constructs that we've seen, why is it that natural language can't be compiled like a programming language? After all, it would be difficult to program if a compiler performed as poorly as Microsoft Word's grammar checker.
- There are a number of technical difficulties in writing a language parser. Notably, there are many times where nodes point to words that are phonologically null, or do not exist depending on the context.

Problems with Syntax Parsing

- With these constructs that we've seen, why is it that natural language can't be compiled like a programming language? After all, it would be difficult to program if a compiler performed as poorly as Microsoft Word's grammar checker.
- There are a number of technical difficulties in writing a language parser. Notably, there are many times where nodes point to words that are phonologically null, or do not exist depending on the context.
- Additionally, words move around in the tree, such as in the case of Subject-Aux Inversion, and tense modifiers for verbs attach at a different point in the tree.

More Problems with Language Processing

- However, these seem to be difficulties in implementation — our algorithm is sufficiently general that we should be able to modify it so it can match nonterminals to the empty strings, or reorder words on the tree.

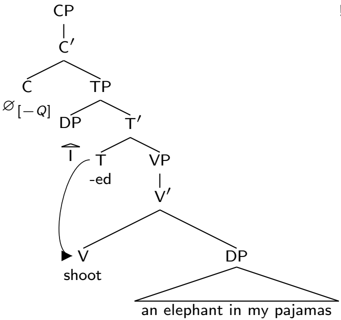
More Problems with Language Processing

- However, these seem to be difficulties in implementation — our algorithm is sufficiently general that we should be able to modify it so it can match nonterminals to the empty strings, or reorder words on the tree.
- The more significant problems is that natural language is inherently ambiguous; Groucho Marx's quip "I shot an elephant in my pajamas" comes to mind.

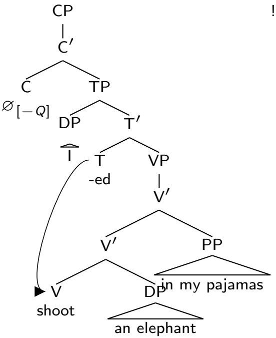
More Problems with Language Processing

- However, these seem to be difficulties in implementation — our algorithm is sufficiently general that we should be able to modify it so it can match nonterminals to the empty strings, or reorder words on the tree.
- The more significant problems is that natural language is inherently ambiguous; Groucho Marx's quip "I shot an elephant in my pajamas" comes to mind.
- Even in syntactically unambiguous sentences, there can exist semantic ambiguities that make the meaning of the sentence unclear.

Syntactic Ambiguities



!



!

Semantic Ambiguities

- Additionally, there are ambiguities that go beyond the syntactic structure. Consider the following pair of sentences:

Semantic Ambiguities

- Additionally, there are ambiguities that go beyond the syntactic structure. Consider the following pair of sentences:
    - "We gave [the monkeys]$_i$ [the bananas]$_j$ because [they]$_i$ were hungry."

Semantic Ambiguities

- Additionally, there are ambiguities that go beyond the syntactic structure. Consider the following pair of sentences:
  - “We gave [the monkeys]$_i$ [the bananas]$_j$ because [they]$_i$ were hungry.”
  - “We gave [the monkeys]$_i$ [the bananas]$_j$ because [they]$_j$ were ripe.”

Semantic Ambiguities

- Additionally, there are ambiguities that go beyond the syntactic structure. Consider the following pair of sentences:
  - "We gave [the monkeys]$_i$ [the bananas]$_j$ because [they]$_i$ were hungry."
  - "We gave [the monkeys]$_i$ [the bananas]$_j$ because [they]$_j$ were ripe."
- While these two sentences have identical syntax trees, the pronoun "they" is *bound* to two different phrases in the two different sentences. (SpecGram)

More Semantic Ambiguities

- Sometimes, even humans can't resolve semantic ambiguities. Consider the following three sentences:

More Semantic Ambiguities

- Sometimes, even humans can't resolve semantic ambiguities. Consider the following three sentences:
  - [John]$_i$ told [Jack]$_j$ that [he]$_i$ needed a break.

More Semantic Ambiguities

- Sometimes, even humans can't resolve semantic ambiguities. Consider the following three sentences:
  - [John]$_i$ told [Jack]$_j$ that [he]$_i$ needed a break.
  - [John]$_i$ told [Jack]$_j$ that [he]$_j$ needed a break.

More Semantic Ambiguities

- Sometimes, even humans can't resolve semantic ambiguities. Consider the following three sentences:
  - [John]$_i$ told [Jack]$_j$ that [he]$_i$ needed a break.
  - [John]$_i$ told [Jack]$_j$ that [he]$_j$ needed a break.
  - [John]$_i$ told [Jack]$_j$ that [he]$_k$ needed a break.

More Semantic Ambiguities

- Sometimes, even humans can't resolve semantic ambiguities. Consider the following three sentences:
  - [John]$_i$ told [Jack]$_j$ that [he]$_i$ needed a break.
  - [John]$_i$ told [Jack]$_j$ that [he]$_j$ needed a break.
  - [John]$_i$ told [Jack]$_j$ that [he]$_k$ needed a break.
- Any of these bindings can be correct.

So What Do We Do Now?

- Problems like the ones just mentioned are known as *NLP-hard* problems; that is, solving these problems requires at least a program capable of fully understanding natural language.

So What Do We Do Now?

- Problems like the ones just mentioned are known as *NLP-hard* problems; that is, solving these problems requires at least a program capable of fully understanding natural language.
- Of course, this doesn't stop researchers from trying to crack the problems of language.

So What Do We Do Now?

- Problems like the ones just mentioned are known as *NLP-hard* problems; that is, solving these problems requires at least a program capable of fully understanding natural language.
- Of course, this doesn't stop researchers from trying to crack the problems of language.
- There are quite a number of subproblems, other areas of research, and other interesting things to study when it comes to natural language.

Probabilistic Grammars

- We have seen that there exist sentences for which there exist multiple parse trees. With longer sentences there could be quite a large number of derivations.

Probabilistic Grammars

- We have seen that there exist sentences for which there exist multiple parse trees. With longer sentences there could be quite a large number of derivations.
- Therefore, instead of having a purely deterministic grammar, we can introduce a grammar that has probabilities associated with each production.

Probabilistic Grammars

- We have seen that there exist sentences for which there exist multiple parse trees. With longer sentences there could be quite a large number of derivations.
- Therefore, instead of having a purely deterministic grammar, we can introduce a grammar that has probabilities associated with each production.
- Then we can view the space of all possible parses as a search space, and use Prolog to search and determine the relative probabilities of each interpretation.

Probabilistic Grammars

- We have seen that there exist sentences for which there exist multiple parse trees. With longer sentences there could be quite a large number of derivations.
- Therefore, instead of having a purely deterministic grammar, we can introduce a grammar that has probabilities associated with each production.
- Then we can view the space of all possible parses as a search space, and use Prolog to search and determine the relative probabilities of each interpretation.
- The most likely syntax tree is chosen as the correct parsing of that sentence.

Probabilistic Grammars

- We have seen that there exist sentences for which there exist multiple parse trees. With longer sentences there could be quite a large number of derivations.
- Therefore, instead of having a purely deterministic grammar, we can introduce a grammar that has probabilities associated with each production.
- Then we can view the space of all possible parses as a search space, and use Prolog to search and determine the relative probabilities of each interpretation.
- The most likely syntax tree is chosen as the correct parsing of that sentence.
- These parsers are often supplemented with corpuses (large bodies of text) that can aid in computing the probabilities of each rule.

Analyzing Human Productions of Speech

- Of course, language is not always given to parsers in neatly formatted ASCII files.

Analyzing Human Productions of Speech

- Of course, language is not always given to parsers in neatly formatted ASCII files.
- It would if parsers could interface with existing language samples. By and large, this means speech recognition, although handwriting recognition has received some attention as well.

Analyzing Human Productions of Speech

- Of course, language is not always given to parsers in neatly formatted ASCII files.
- It would if parsers could interface with existing language samples. By and large, this means speech recognition, although handwriting recognition has received some attention as well.
- Speech recognition is still certainly an open problem — programs can recognize a small number of words spoken by a large number of people, or a large number of words spoken by a small number of people, but not both.

Analyzing Human Productions of Speech

- Of course, language is not always given to parsers in neatly formatted ASCII files.
- It would if parsers could interface with existing language samples. By and large, this means speech recognition, although handwriting recognition has received some attention as well.
- Speech recognition is still certainly an open problem — programs can recognize a small number of words spoken by a large number of people, or a large number of words spoken by a small number of people, but not both.
- Anyone who has seen the infamous Windows Vista speech recognition demo knows that there is a lot that can go wrong.

Hidden Markov Models

- A Markov model is a finite-state automaton where each state transition has a certain probability associated with it.

Hidden Markov Models

- A Markov model is a finite-state automaton where each state transition has a certain probability associated with it.
- We can model speech as a Markov model where each state represents a phoneme and the transitions are weighted with the likelihood to move from state to state, but the probabilistic parameters are unknown.

Hidden Markov Models

- A Markov model is a finite-state automaton where each state transition has a certain probability associated with it.
- We can model speech as a Markov model where each state represents a phoneme and the transitions are weighted with the likelihood to move from state to state, but the probabilistic parameters are unknown.
- From there, we can feed the model sets of preworked data, from which it can build the probabilities.

Hidden Markov Models

- A Markov model is a finite-state automaton where each state transition has a certain probability associated with it.
- We can model speech as a Markov model where each state represents a phoneme and the transitions are weighted with the likelihood to move from state to state, but the probabilistic parameters are unknown.
- From there, we can feed the model sets of preworked data, from which it can build the probabilities.
- Intuitively, this makes sense — certain phoneme sequences like [kn] are just not very likely, and we can exploit that to eliminate those types of possibilities *en masse*. (Rabiner)

Automatic Tranlations

- Anyone who has used a computer to translate something in a foreign language has seen that it produces less-than-desirable results that are frequently not symmetric.

Automatic Tranlations

- Anyone who has used a computer to translate something in a foreign language has seen that it produces less-than-desirable results that are frequently not symmetric.
- For example, Google translates “John drove to the store in his car after school today.” into Italian as “Giovanni ha spinto a conservare nella sua auto dopo la scuola di oggi.” In turn, that is translated back into English as “John has pushed to keep in his car after school today.”

Automatic Tranlations

- Anyone who has used a computer to translate something in a foreign language has seen that it produces less-than-desirable results that are frequently not symmetric.
- For example, Google translates "John drove to the store in his car after school today." into Italian as "Giovanni ha spinto a conservare nella sua auto dopo la scuola di oggi." In turn, that is translated back into English as "John has pushed to keep in his car after school today."
- Obviously this is one of the harder problems facing NLP, and while there have been a number of commmercial efforts, none have been particularly successful.

Challenges of Automatic Translation

- One of the major challenges is, as mentioned above, languages are inherently ambiguous. This causes problems because each language has a different set of ambiguities.

Challenges of Automatic Translation

- One of the major challenges is, as mentioned above, languages are inherently ambiguous. This causes problems because each language has a different set of ambiguities.
- For example, the English wordplay "Time flies like an arrow, fruit flies like a banana." would not be ambiguous in Japanese because it is a SOV language - that is, its verbs go on the ends of sentences, and the subject and object are marked by "-ga" and "-o" respectively.

Challenges of Automatic Translation

- One of the major challenges is, as mentioned above, languages are inherently ambiguous. This causes problems because each language has a different set of ambiguities.
- For example, the English wordplay "Time flies like an arrow, fruit flies like a banana." would not be ambiguous in Japanese because it is a SOV language - that is, its verbs go on the ends of sentences, and the subject and object are marked by "-ga" and "-o" respectively.
- In other cases, information is lost in translation. Romance languages are especially notorious for omitting subject pronouns before verbs, which causes gender to be lost.

Challenges of Automatic Translation

- One of the major challenges is, as mentioned above, languages are inherently ambiguous. This causes problems because each language has a different set of ambiguities.
- For example, the English wordplay "Time flies like an arrow, fruit flies like a banana." would not be ambiguous in Japanese because it is a SOV language - that is, its verbs go on the ends of sentences, and the subject and object are marked by "-ga" and "-o" respectively.
- In other cases, information is lost in translation. Romance languages are especially notorious for omitting subject pronouns before verbs, which causes gender to be lost.
- Because of these difficulties, machine translation most likely requires a tool capable of completely understanding a language.

News and Blog Analysis with Lydia

- Of course, the professors of Stony Brook have made their own contributions to natural language processing.

News and Blog Analysis with Lydia

- Of course, the professors of Stony Brook have made their own contributions to natural language processing.
- The project has received contributions from Mikhail Bautin, Anand Mallangada, Alex Turner, Lohit Vijaya-renu, and Steven Skiena and others

News and Blog Analysis with Lydia

- Of course, the professors of Stony Brook have made their own contributions to natural language processing.
- The project has received contributions from Mikhail Bautin, Anand Mallangada, Alex Turner, Lohit Vijaya-renu, and Steven Skiena and others
- The goals of Lydia include the automatically spidering of news stories and their correlate them based on common terms and ideas, as well as identifying subjectivity in news stories.

Analyzing Subjectivity

- One interesting use of Lydia is analyzing subjectivity towards a particular concept from different languages. To do this, a quality machine translator was needed; IBM's Websphere Translation Server proved to be sufficient to their purpose.

Analyzing Subjectivity

- One interesting use of Lydia is analyzing subjectivity towards a particular concept from different languages. To do this, a quality machine translator was needed; IBM's Websphere Translation Server proved to be sufficient to their purpose.
- Then a wordmap of positive and negative adjectives could be built. For each day, the number of positive and negative adjectives concerning a particular concept (in this case Korea) was counted.

Analyzing Subjectivity

- Finally, for a given language l, day d, and entity e, the subjectivity could be calculated as:

$$\frac{pos\_references_{l,e,d} - neg\_references_{l,e,d}}{num_occurences\_l, e, d}$$

Analyzing Subjectivity

- Finally, for a given language l, day d, and entity e, the subjectivity could be calculated as:

$$\frac{pos\_references_{l,e,d} - neg\_references_{l,e,d}}{num_o ccurences\_l, e, d}$$

- The results of running this experiment for a week returned expected results; Korean had the highest subjectivity score, while Japan was among the lowest, and the several European countries sampled were in the middle. (Bautin et al)