# A survey of parsing and its applications

Mark Johnson

Macquarie University
Sydney, Australia

August 2015

# Outline

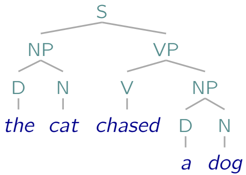
# What's driving NLP and CL research?

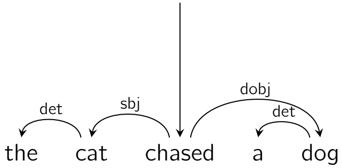
- Tools for managing the "information explosion"
  - extracting information from and managing large text document collections
  - NLP is often free "icing on the cake" to sell more ads;
    e.g., speech recognition, machine translation, document clustering (news), etc.
- *Mobile and portable computing*
  - keyword search / document retrieval don't work well on very small devices
  - we want to be able to talk to our computers (speech recognition)
    and have them say something intelligent back (question-answering, generation)
- The intelligence agencies
- The old Artificial Intelligence (AI) dream
  - *language is the most direct window into the mind*

# Different kinds of linguistic regularities

- *Phonology* studies the *distributional patterns of sounds*
  - E.g., *cats* vs *dogs*
- *Morphology* studies the *structure of words*
  - E.g., *re+vital+ise*
- *Syntax* studies how *words combine to form phrases and sentences*
  - E.g., *I saw the man with the telescope*
- *Semantics* studies how *language conveys meaning*
  - E.g., *I sprayed the paint onto the wall/I sprayed the wall with paint*
- *Pragmatics* studies how *language is used to do things*
  - E.g., *Can you pass the salt?*

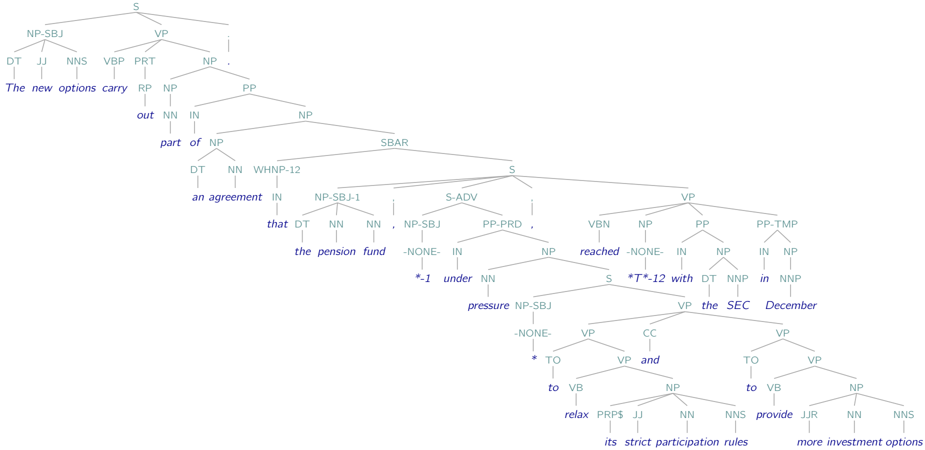
# Phrase structure and dependency parses




- A *phrase structure parse* represents phrases as nodes in a tree
- A *dependency parse* represents dependencies between words
- Phrase structure and dependency parses are *approximately inter-translatable*:
  - dependency structures assume *all phrases have a unique head*
  - ⇒ phrase structure can describe a wider range of syntactic constructions than dependency representations
- Phrase structure parsing was studied in depth before dependency parsing
- Phrase structure parsing is typically slower (tens of sentences/sec) than dependency parsing (thousands of sentences/sec)

# Syntactic parses of real sentences




- State-of-the-art parsers have accuracies of over 90%

⇒ *Most parses contain at least one error*

# Advantages of probabilistic parsing

- In the GofAI approach to syntactic parsing:
  - a hand-written grammar defines the grammatical (i.e., well-formed) parses
  - given a sentence, the parser returns the set of grammatical parses for that sentence
  - ⇒ unable to *distinguish more likely from less likely parses*
  - ⇒ hard to ensure *robustness* (i.e., that every sentence gets a parse)
- In a probabilistic parser:
  - the grammar *generates all possible parse trees* for all possible strings (roughly)
  - use probabilities to identify plausible syntactic parses
- Probabilistic syntactic models usually encode:
  - the probabilities of syntactic constructions
  - the probabilities of lexical dependencies
    e.g., how likely is *pizza* as direct object of *eat*?

# Probabilistic Context-Free Grammars

- Probabilistic context-free grammars (PCFGs) define *probability distributions over trees*
- Each *nonterminal node* expands by
  - choosing a rule expanding that nonterminal, and
  - recursively expanding any nonterminal children it contains
- Probability of tree is *product of probabilities of rules* used to construct it

| *Probability* $\theta_r$ | *Rule r* |
|---|---|
| 1 | S → NP VP |
| 0.7 | NP → *Sam* |
| 0.3 | NP → *Sandy* |
| 1 | VP → V NP |
| 0.8 | V → *likes* |
| 0.2 | V → *hates* |

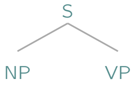
S

$$\mathsf{P}(\text{Tree}) \quad =$$

# Probabilistic Context-Free Grammars

- Probabilistic context-free grammars (PCFGs) define *probability distributions over trees*
- Each *nonterminal node* expands by
  - choosing a rule expanding that nonterminal, and
  - recursively expanding any nonterminal children it contains
- Probability of tree is *product of probabilities of rules* used to construct it

| *Probability* $\theta_r$ | *Rule r* |
|---|---|
| 1 | S → NP VP |
| 0.7 | NP → *Sam* |
| 0.3 | NP → *Sandy* |
| 1 | VP → V NP |
| 0.8 | V → *likes* |
| 0.2 | V → *hates* |

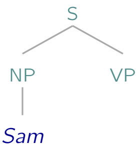



$$P(\text{Tree}) = 1 \times$$

# Probabilistic Context-Free Grammars

- Probabilistic context-free grammars (PCFGs) define *probability distributions over trees*
- Each *nonterminal node* expands by
  - choosing a rule expanding that nonterminal, and
  - recursively expanding any nonterminal children it contains
- Probability of tree is *product of probabilities of rules* used to construct it

| *Probability* $\theta_r$ | *Rule* $r$ |
|---|---|
| 1 | S → NP VP |
| 0.7 | NP → *Sam* |
| 0.3 | NP → *Sandy* |
| 1 | VP → V NP |
| 0.8 | V → *likes* |
| 0.2 | V → *hates* |

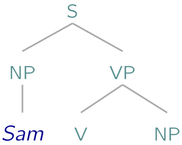



$$\mathsf{P}(\text{Tree}) = 1 \times 0.7 \times$$

# Probabilistic Context-Free Grammars

- Probabilistic context-free grammars (PCFGs) define *probability distributions over trees*
- Each *nonterminal node* expands by
  - choosing a rule expanding that nonterminal, and
  - recursively expanding any nonterminal children it contains
- Probability of tree is *product of probabilities of rules* used to construct it

| *Probability $\theta_r$* | *Rule $r$* |
|---|---|
| 1 | S $\rightarrow$ NP VP |
| 0.7 | NP $\rightarrow$ *Sam* |
| 0.3 | NP $\rightarrow$ *Sandy* |
| 1 | VP $\rightarrow$ V NP |
| 0.8 | V $\rightarrow$ *likes* |
| 0.2 | V $\rightarrow$ *hates* |

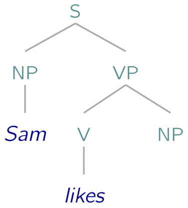



$$\mathsf{P}(\text{Tree}) \quad = \quad 1 \times 0.7 \times 1 \times$$

# Probabilistic Context-Free Grammars

- Probabilistic context-free grammars (PCFGs) define *probability distributions over trees*
- Each *nonterminal node* expands by
  - choosing a rule expanding that nonterminal, and
  - recursively expanding any nonterminal children it contains
- Probability of tree is *product of probabilities of rules* used to construct it

| *Probability* $\theta_r$ | *Rule r* |
|---|---|
| 1 | S → NP VP |
| 0.7 | NP → *Sam* |
| 0.3 | NP → *Sandy* |
| 1 | VP → V NP |
| 0.8 | V → *likes* |
| 0.2 | V → *hates* |

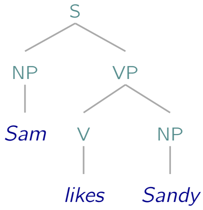



$$\mathsf{P}(\text{Tree}) \quad = \quad 1 \times 0.7 \times 1 \times 0.8 \times$$

# Probabilistic Context-Free Grammars

- Probabilistic context-free grammars (PCFGs) define *probability distributions over trees*
- Each *nonterminal node* expands by
  - choosing a rule expanding that nonterminal, and
  - recursively expanding any nonterminal children it contains
- Probability of tree is *product of probabilities of rules* used to construct it

| *Probability* $\theta_r$ | *Rule* $r$ |
|---|---|
| 1 | S $\rightarrow$ NP VP |
| 0.7 | NP $\rightarrow$ *Sam* |
| 0.3 | NP $\rightarrow$ *Sandy* |
| 1 | VP $\rightarrow$ V NP |
| 0.8 | V $\rightarrow$ *likes* |
| 0.2 | V $\rightarrow$ *hates* |




$$\mathsf{P}(\text{Tree}) \quad = \quad 1 \times 0.7 \times 1 \times 0.8 \times 0.3$$

# Two uses for probabilistic syntactic parsing

- A probabilistic *syntactic parser* returns *a list of syntactic parses together with their probabilities* for each sentence

⇒ Use *most probable parse* to help understand the sentence

  - question answering
  - *information extraction*

⇒ Use the *sum of parse probabilities* to estimate the probability of a sentence (*syntactic language model*)

  - speech recognition
  - machine translation
  - *speech error detection and correction*

# Outline

# A typology of speech disfluencies

- *Filled pauses:*

  I think it's uh refreshing to see the uh support . . .

- *Parentheticals*

  But you know I was reading the other day . . .

- *Repairs:*

  I want a flight to Boston uh I mean to Denver on Friday

- *Restarts:*

  Why didn't he why didn't she stay at home?

Bear, Dowding and Schriberg (1992), Heeman and Allen (1997, 1999), Nakatani and Hirschberg (1994), Stolcke and Schriberg (1996)

# Why treat restarts and repairs specially?

- Filled pauses are easy to recognise and remove *from speech transcripts*
- Modern NLP tools (e.g., parsers) handle parentheticals properly
- But *restarts and repairs* are often *misanalysed* by NLP tools

⇒ Detect and remove disfluencies before further processing

*I want a flight to Boston uh I mean to Denver on Friday*
*Why didn't he why didn't she stay at home?*

# The structure of restarts and repairs

. . . and you get, uh, you can get a system . . .

Reparandum: you get, | Interregnum: uh, | Repair: you can get

- The Reparandum is *often not a syntactic phrase*
- The Interregnum is usually lexically and prosodically marked, but can be empty
- The Reparandum is often a *"rough copy"* of the Repair
  - Repairs are typically short
  - Repairs are not always copies
  - It's possible e.g. for there to be anaphoric dependencies into the reparandum

Shriberg 1994 "Preliminaries to a Theory of Speech Disfluencies"

# Machine-learning approaches to disfluency detection

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | _ | E | E | E | _ | _ | _ | _ | _ | |
| … | and | you | get | uh | you | can | get | a | system | … |
| | | Reparandum | | Interregnum | Repair | | | | | |

- Train a classifier to predict whether each word is Edited or NotEdited
  - this approach classifies each word independently, but the classification should really be made over groups of words
- A *very large number of features* can be usefully deployed in such a system

Charniak and Johnson (2001), Zhang, Weng and Feng (2006)

# The “true” model of repairs (?)

. . . and $\underbrace{\text{you get,}}_{\text{Reparandum}}$ $\underbrace{\text{uh,}}_{\text{Interregnum}}$ $\underbrace{\text{you can get}}_{\text{Repair}}$ a system . . .

- Speaker generates intended “conceptual representation”
- Speaker incrementally generates syntax and phonology,
  - recognizes that what is said doesn’t mean what was intended,
  - “backs up”, i.e., partially deconstructs syntax and phonology, and
  - starts incrementally generating syntax and phonology again
- (but without a good model of “conceptual representation”, this may be hard to formalize . . . )

# Approximating the "true model"

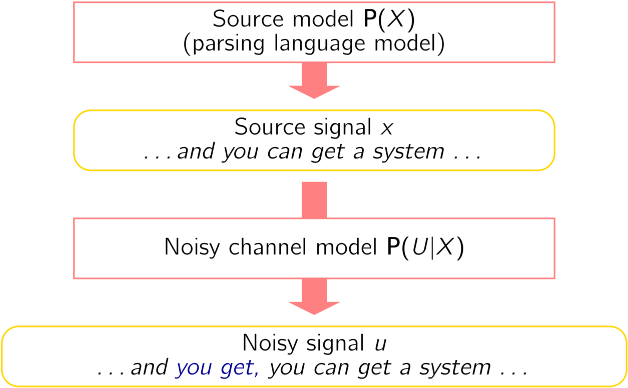
I want a flight to Boston uh I mean to Denver on Friday

- Use Repair string as approximation to intended meaning
- Reparandum string is "rough copy" of Repair string
  - involves *crossing* (rather than *nested*) dependencies
- String with reparandum and interregnum excised is usually well-formed
  - after correcting the error, what's left should have high probability

  ⇒ *use model of normal language to interpret ill-formed input*
- A parsing model can check that the proposed repaired string is *grammatically well-formed*
  - speech errors tend to occur at the beginnings of clauses and major phrases

  ⇒ use parsing model to check that speech errors occur in syntactically plausible locations

# The Noisy Channel Model




- Noisy channel models combines two different submodels
- Channel model needs to generate *crossing dependencies*
  ⇒ TAG transducer

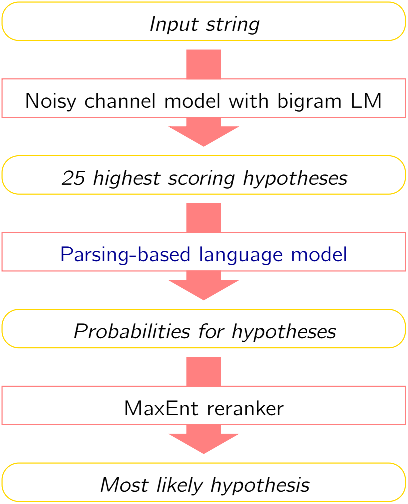
Johnson and Charniak (2004)

# Reranking the Noisy Channel model

- Log probs from source model and channel model are *reranker features*
- MaxEnt reranker can use *additional features* as well

⇒ Best of both noisy channel and machine-learning approaches

- Johnson et al used a parser-based language model



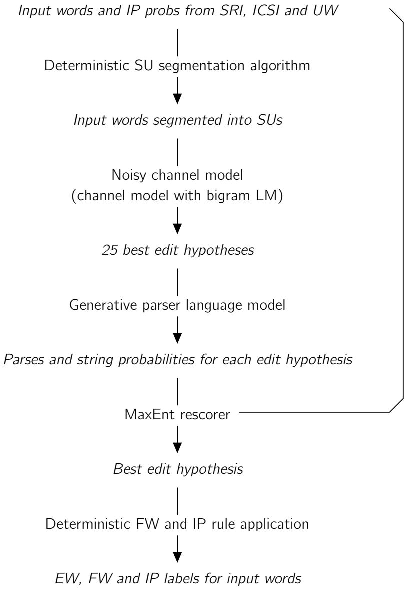
# Evaluation of model's performance

| | f-score | error rate |
|---|---|---|
| NCM + bigram LM | 0.75 | 0.45 |
| NCM + parser LM | 0.81 | 0.35 |
| MaxEnt rescorer using NCM + parser LM | 0.87 | 0.25 |
| MaxEnt rescorer alone | 0.78 | 0.38 |

- Evaluated on unseen portion of Switchboard corpus
- *f-score* is a geometric average of Edited words precision and recall (bigger is better)
- *error rate* is the number of Edited word errors made divided by number of true edited words (smaller is better)

# RT04F competition

- RT04F evaluated *meta-data extraction*
    - disfluency detection/correction was just one of the tasks they evaluated
- Test material was unsegmented speech recognizer output or transcripts
- ICSI, SRI and UW supplied us with ASR output, SU boundaries and acoustic IP probabilities
- Added rescorer features that incorporated these
- *Won all of the RT04F disfluency detection competitions we entered*



# Further results on disfluency detection/correction

- Zwarts, Johnson and Dale (2010) developed an *incremental version of this algorithm* for detecting and correcting speech repairs
- Zwarts and Johnson (2011) evaluate *the effect of language model choice* on disfluency detection and correction
  - parsing-based language models do better than n-gram models, all else equal
  - best performance comes from combining all the language models in a single model
- Honnibal and Johnson (2014) present *a joint model of dependency parsing and disfluency detection/correction*
  - we augment the shift-reduce actions of a *transition-based dependency parser* with a special *detach action* that "disconnects" a word or partial phrase from the parse tree
  - this model is inherently incremental
  - because the parse tree is constructed at the same time, it's easy to exploit syntactic structure for detecting speech disfluencies
  - lead to work on *non-monotonic transition-based parsing algorithms* which use specialised "repair transitions" to *correct earlier parsing errors*

# Outline

# Named entity recognition and linking

- *Named entity recognition* finds all "mentions" referring to an entity in a document

  Example: $\underbrace{\textit{Tony Abbott}}_{\text{person}}$ *bought* $\underbrace{\textit{300}}_{\text{number}}$ *shares in* $\underbrace{\textit{Acme Corp}}_{\text{corporation}}$ *in* $\underbrace{\textit{2006}}_{\text{date}}$

- *Noun phrase coreference* tracks mentions to entities within or across documents

  Example: *Tony Abbott met the president of Indonesia yesterday. Mr. Abbott told him that he ...*

- *Entity linking* maps entities to database entries

  Example: $\underbrace{\textit{Tony Abbott}}_{\text{/m/xw2135}}$ *bought* $\underbrace{\textit{300}}_{\text{number}}$ *shares in* $\underbrace{\textit{Acme Corp}}_{\text{/m/yzw9w}}$ *in* $\underbrace{\textit{2006}}_{\text{date}}$

# Relation extraction

- *Relation extraction* mines texts to find *relationships between named entities*, i.e., "who did what to whom (when)?"

  *The new Governor General, Peter Cosgrove, visited Buckingham Palace yesterday.*

  **Has-role**

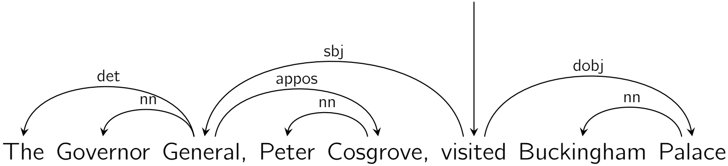
  | **Person** | **Role** |
  |---|---|
  | Peter Cosgrove | Governor General of Australia |

  **Offical-visit**

  | **Visitor** | **Organisation** |
  |---|---|
  | Peter Cosgrove | Queen of England |

- The syntactic parse provides useful features for relation extraction
- Bio-medical research literature and financial documents are major application areas
- Ignores lots of potentially relevant information, e.g., *yesterday*

# Syntactic parsing for relation extraction

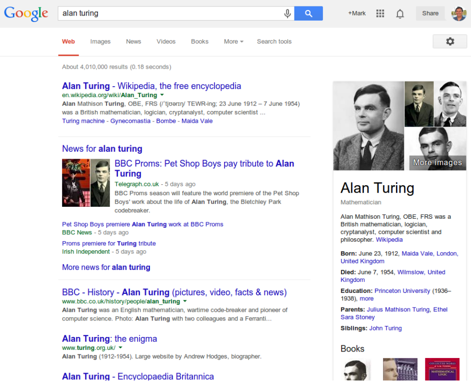



- The *syntactic path* in a *dependency parse* is a useful feature in relation extraction

$$X \xrightarrow{\text{appos}} Y \Rightarrow \textit{has-role}(Y, X)$$

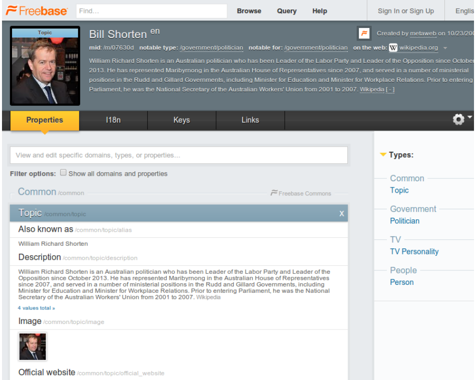
$$X \xleftarrow{\text{sbj}} \textit{visited} \xrightarrow{\text{dobj}} Y \Rightarrow \textit{official-visit}(X, Y)$$

# Google’s Knowledge Graph




- Goal: move beyond keyword search document retrieval to *directly answer user queries*
  - ⇒ easier for mobile device users
- Google’s Knowledge Graph:
  - built on top of FreeBase
  - entries are synthesised from Wikipedia, news stories, etc.
  - manually curated (?)

# FreeBase: an open knowledge base




- An entity-relationship database on top of a graph triple store
- Data mined from Wikipedia, ChefMoz, NNDB, FMD, MusicBrainz, etc.
- 44 million topics (entities), 2 billion facts, 250GB uncompressed dump
- Created by Metaweb, which was acquired by Google

# Distant supervision for relation extraction

- Ideal labelled data for relation extraction: large text corpus annotated with entities and relations
  - expensive to produce, especially for a lot of relations!
- *Distant supervision assumption:* if two or more entities that appear in the same sentence also appear in the same database relation, then probably the sentence expresses the relation
  - assumes *pairs of entities only interact in one way*
  - *temporal information* can resolve some ambiguities
- With the distant supervision assumption, we obtain relation extraction training data by:
  - taking a large text corpus (e.g., 10 years of news articles)
  - running a named entity linker on the corpus
  - looking up the entity tuples that appear in the same sentence in the large knowledge base (e.g., FreeBase)

⇒ Enables us to learn parsing-based extraction patterns for each FreeBase relation

# Opinion mining and sentiment analysis

- Used to analyse e.g., social media (Web 2.0)
- Typical goals: given a corpus of messages:
  - classify each message along a *subjective–objective scale*
  - identify the message *polarity* (e.g., on dislike–like scale)
- Training opinion mining and sentiment analysis models:
  - in some domains, *supervised learning* with simple *keyword-based features* works well
  - but in other domains it's necessary to model *syntactic structure* as well
    - E.g., *I doubt she had a very good experience ...*
- Opinion mining can be combined with:
  - *topic modelling* to cluster messages with similar opinions
  - multi-document *summarisation* to summarise results
  - *named entity linking* and *relation extraction* to associate sentiment with specific entities (e.g., *I like Windows much more than Linux*).

# Outline

# Which Jim Jones?

- News text: *Jim Jones' recent musical releases . . .*
- 8 Wikipedia pages for *Jim Jones*:
  - 2 politicians
  - 1 basketball player
  - 1 hockey player
  - 1 guitarist (deceased)
  - 1 rapper
  - 1 cult leader (deceased)
- *How do we know it's the rapper?*

# Life Stories

- A person's *life story* is the sequence of events that occur to them
- Generalisations about life stories:
  - everyone dies less than 110 years after they were born
  - if someone goes to school,
    it's usually when they are 5–20 years old
  - if someone goes to college,
    it's often immediately after school
  - a singer is more likely than a carpenter
    to have a musical release
  - an academic is more likely than an accountant
    to write a book
  - a lawyer is more likely than an actor
    to become a politician

# The structure of life stories

- Everybody's life story is different
  ⇒ finite set of "life templates" won't suffice
- But there are generalisations:
  - few artists have exactly 10 CDs like Jim Jones
  - but releasing a CD is a frequent event for artists like Jim Jones, with predictable subevents:
    - release parties
    - promotions and reviews
    - shows and tours
- *Can we learn typical life stories?*
- *Given a partial life story, can we "fill in" the rest?*

# Life Stories and Topic Models

| LDA topic models | Life story models |
|---|---|
| *words* | *events* (e.g., running for election, releasing a CD) |
| *documents* | *life stories* (the sequence of events in an individual's life) |
| *topics* | *careers* (sequences of events associated with e.g., being a politician or musician) |

- Topics are hidden when training a topic model, while FreeBase has abundant information about events
  - identifying the *relevant information* may be hard

# What are Life Stories?

- FreeBase as a repository of Life Stories
  - FreeBase contains more than 100 properties for $\approx$ 250,000 people
  - Coverage is uneven: Sarah Palin's political career is covered, her political commentator roles on Fox News are not
- What appears in a Life Story?
  - time-stamped properties, e.g., *Bill Clinton's presidency 1993–2001*
  - indirectly time-stamped properties, e.g., *Bill Clinton's 1996 presidential campaign*
  - some properties without timestamps, e.g., *gender, nationality, notable type*
- Possible formalisations of Life Stories
  - temporally-bounded sets of events (i.e., a time-line)
  - events occuring in fixed windows (e.g., each year's events)

# Applications of Life Stories

- *Disambiguating named entities and relations* in automatic knowledge extraction
  - *bias syntactic parsing and semantic interpretation toward plausible relationships*
  - help disambiguate named entities
- *Error and anomaly detection:*
  - highly improbable clusters of events (e.g., someone simultaneously being an astronaut and a sportsperson) may indicate errors in the knowledge base
- *Fraud detection:*
  - highly improbable sequences of events might not have actually happened
- *Discovering unusual individuals:*

# Important events

- Events differ in importance
  - Bill Clinton made 97 political appointments,
    appeared on 24 TV shows, and
    was elected US President twice
- FreeBase internal measures of importance
  - *causes* are highly predictive, temporally-preceding event types
- External measures of importance or impact
  - use relation extraction to align FreeBase properties to the individual's Wikipedia text, or a large news corpus
  - estimate importance by *amount of text* (sentences, column inches, etc.) linked to event

# Event structure

- Events have a complicated *temporal* and *causal* structure
  - Bill Clinton's winning the 1996 Presidental election

  $\Rightarrow$ Bill Clinton is US President 1997–2001

  $\Rightarrow$ Bill Clinton makes 97 political appointments
- At what *granularity* should we individuate events?
  Many useful tasks don't require detailed information
  - dead cult leaders don't release hit CDs
- Minor events can give information about important events
  - a late alimony payment $\Rightarrow$ marriage and divorce
- Can *hierarchical models* generalise at multiple levels simultaneously?

# Evaluating a Life Story model

- Life Story models should be useful in
  - named entity linking
  - relation extraction

  but accuracy on those tasks depends on other factors as well
- Evaluate the predictive ability of a Life Story model, e.g.:
  - train model on 2012 FreeBase
  - give model an individual's pre-2013 Life Story
    and several possible 2013 completions
  - evaluate how accurately model chooses correct completion

# Example: Dick Cheney

## The story until 2000

- born 1941, in Lincoln, Nebraska
- studied political science at the University of Nebraska
- White House chief of staff 1975–1977
- elected to US Congress 1979–1989
- minority whip in US Congress 1989
- US Secretary for Defense 1989–1993
- employed by Halliburton 1995–2000

## 2001 alternative #1

- litigant in Supreme Court legal case
- Vice President of the United States
- founded Energy Task Force

## 2001 alternative #2

- mayor of Wasilla, Alaska
- member of the Alaska Municipal League board

# Life Story models

- The future is like the past, i.e., choose the completion which is as close as possible to the known events
- Binary classifier that predicts how likely the future events are given the past events
  - can learn simple contextual generalisations
    e.g., an academic is more likely to write a book than a sportsperson
- *n*-gram and Hidden Markov Models
  - linearize events into a sequence
  - project events onto a finite set of event types
- Hierarchical models of Life Stories
  - a Life Story is a (possibly overlapping) sequence of *careers*
  - each *career* is a sequence of *events*
  - each *event* has *properties* and a *duration*

# Outline

# Summary

- Because semantics is generally *compositional*, recovering syntactic structure is a *key step in understanding the meaning of a sentence*
- There are two popular kinds of syntactic representation: *phrase structures* and *dependency structures*
  - phrase structures can describe a wider range of syntactic constructions
  - dependency structures are faster and easier to produce
- *Probabilistic parsing models* compute possible parses for a sentence, together with their probabilities
  - ⇒ parsing models can be used as *syntactic language models* to distinguish plausible from implausible sentences
    - we've used them to consistently develop the best disfluency-detection systems for over a decade
  - ⇒ parsing models can be used to identify the *most plausible syntactic analysis of a sentence*
    - parsing plays a key role in information extraction systems
- The best syntactic parsers have around 92% accuracy ⇒ *most parses contain an error*
  - ⇒ *there's still much more work to do on syntactic parsing*

# Other research strengths: Topic models

- *Topic models* (e.g., Latent Dirichlet Allocation) are a popular tool for managing large document collections
  - they cluster documents by the words they contain, and cluster words by the topics they appear in
  - they are *bag-of-words models*
- Du, Buntine and Johnson (2013) generalised LDA to *segment documents into topically-coherent parts*
- Du, Pate and Johnson (2015) showed how to *learn topical ordering regularities in a document collection* and use this to improve *document segmentation* and *topic identification*
- Nguyen, Billingsley, Du and Johnson (2015) used *latent word vector representations* learnt from a large external corpus to *improve topic modelling performance on small, specialised document collections* such as Twitter documents
- Zhao, Du, Börschinger, Pate, Ciaramita, Steedman and Johnson (2015) generalises LDA beyond bag-of-words to learn *topical collocations* (e.g., *White House, neural net*)

# Future research: multi-word expressions and parsing

- *Multi-word expressions* appear in many technical technical texts
  - because of sparse data problems, they are often incorrectly parsed
  - often specialised sequence models (e.g., CRFs) are used to recognise them
- Our plan is to *add specialised transitions to a transition-based dependency parser* to detect and parse multi-word expressions