Chapter 15

Reference Resolution

References are one of the most noticeable forms of linguistic ambiguity, afflicting not just automated natural language processing systems, but also fluent human readers. Warnings to avoid "ambiguous pronouns" are ubiquitous in manuals and tutorials on writing style. But referential ambiguity is not limited to pronouns, as shown in the text in Figure 15.1. Each of the bracketed substrings refers to an entity that is introduced earlier in the passage. These references include the pronouns *he* and *his*, but also the shortened name *Cook*, and **nominals** such as *the firm* and *the firm's biggest growth market*.

Reference resolution subsumes several subtasks. This chapter will focus on **coreference resolution**, which is the task of grouping spans of text that refer to a single underlying entity, or, in some cases, a single event: for example, the spans *Tim Cook*, *he*, and *Cook* are all **coreferent**. These individual spans are called **mentions**, because they mention an entity; the entity is sometimes called the **referent**. Each mention has a set of **antecedents**, which are preceding mentions that are coreferent; for the first mention of an entity, the antecedent set is empty. The task of **pronominal anaphora resolution** requires identifying only the antecedents of pronouns. In **entity linking**, references are resolved not to other spans of text, but to entities in a knowledge base. This task is discussed in chapter 17.

Coreference resolution is a challenging problem for several reasons. Resolving different types of **referring expressions** requires different types of reasoning: the features and methods that are useful for resolving pronouns are different from those that are useful to resolve names and nominals. Coreference resolution involves not only linguistic reasoning, but also world knowledge and pragmatics: you may not have known that China was Apple's biggest growth market, but it is likely that you effortlessly resolved this reference while reading the passage in Figure 15.1.[1] A further challenge is that coreference

[1]This interpretation is based in part on the assumption that a **cooperative** author would not use the expression *the firm's biggest growth market* to refer to an entity not yet mentioned in the article (Grice, 1975). **Pragmatics** is the discipline of linguistics concerned with the formalization of such assumptions (Huang,

(15.1) *[[Apple Inc] Chief Executive Tim Cook] has jetted into [China] for talks with government officials as [he] seeks to clear up a pile of problems in [[the firm] 's biggest growth market] ... [Cook] is on [his] first trip to [the country] since taking over...*

Figure 15.1: Running example (Yee and Jones, 2012). Coreferring entity mentions are in brackets.

resolution decisions are often entangled: each mention adds information about the entity, which affects other coreference decisions. This means that coreference resolution must be addressed as a structure prediction problem. But as we will see, there is no dynamic program that allows the space of coreference decisions to be searched efficiently.

15.1 Forms of referring expressions

There are three main forms of referring expressions — pronouns, names, and nominals.

15.1.1 Pronouns

Pronouns are a closed class of words that are used for references. A natural way to think about pronoun resolution is SMASH (Kehler, 2007):

- **S**earch for candidate antecedents;
- **M**atch against hard agreement constraints;
- **A**nd **S**elect using **H**euristics, which are "soft" constraints such as recency, syntactic prominence, and parallelism.

Search

In the search step, candidate antecedents are identified from the preceding text or speech.[2] Any noun phrase can be a candidate antecedent, and pronoun resolution usually requires

2015).

[2]Pronouns whose referents come later are known as **cataphora**, as in the opening line from a novel by Márquez (1970):

(15.1) Many years later, as [he] faced the firing squad, [Colonel Aureliano Buendía] was to remember that distant afternoon when [his] father took him to discover ice.

parsing the text to identify all such noun phrases.[3] Filtering heuristics can help to prune the search space to noun phrases that are likely to be coreferent (Lee et al., 2013; Durrett and Klein, 2013). In nested noun phrases, mentions are generally considered to be the largest unit with a given **head word** (see § 10.5.2): thus, *Apple Inc. Chief Executive Tim Cook* would be included as a mention, but *Tim Cook* would not, since they share the same head word, *Cook*.

Matching constraints for pronouns

References and their antecedents must agree on semantic features such as number, person, gender, and animacy. Consider the pronoun *he* in this passage from the running example:

(15.2) Tim Cook has jetted in for talks with officials as [he] seeks to clear up a pile of problems...

The pronoun and possible antecedents have the following features:

- *he*: singular, masculine, animate, third person
- *officials*: plural, animate, third person
- *talks*: plural, inanimate, third person
- *Tim Cook*: singular, masculine, animate, third person

The SMASH method searches backwards from *he*, discarding *officials* and *talks* because they do not satisfy the agreements constraints.

Another source of constraints comes from syntax — specifically, from the phrase structure trees discussed in chapter 10. Consider a parse tree in which both x and y are phrasal constituents. The constituent x **c-commands** the constituent y iff the first branching node above x also dominates y. For example, in Figure 15.2a, *Abigail* c-commands *her*, because the first branching node above *Abigail*, S, also dominates *her*. Now, if x c-commands y, **government and binding theory** (Chomsky, 1982) states that y can refer to x only if it is a **reflexive pronoun** (e.g., *herself*). Furthermore, if y is a reflexive pronoun, then its antecedent must c-command it. Thus, in Figure 15.2a, *her* cannot refer to *Abigail*; conversely, if we replace *her* with *herself*, then the reflexive pronoun *must* refer to *Abigail*, since this is the only candidate antecedent that c-commands it.

Now consider the example shown in Figure 15.2b. Here, *Abigail* does not c-command *her*, but *Abigail's mom* does. Thus, *her* can refer to *Abigail* — and we cannot use reflexive

[3] In the OntoNotes coreference annotations, verbs can also be antecedents, if they are later referenced by nominals (Pradhan et al., 2011):

(15.1) Sales of passenger cars [grew] 22%. [The strong growth] followed year-to-year increases.

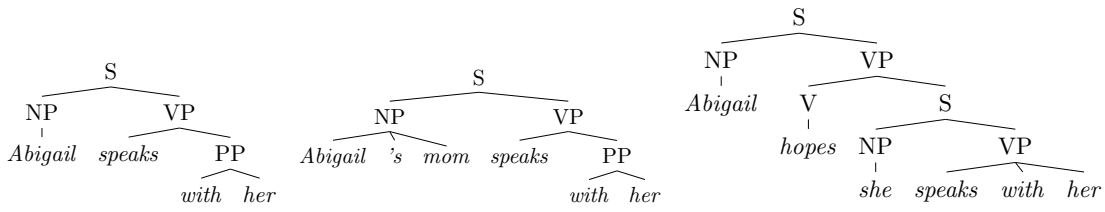

Figure 15.2: In (a), *Abigail* c-commands *her*; in (b), *Abigail* does not c-command *her*, but *Abigail's mom* does; in (c), the scope of *Abigail* is limited by the S non-terminal, so that *she* or *her* can bind to *Abigail*, but not both.

herself in this context, unless we are talking about Abigail's mom. However, *her* does not have to refer to *Abigail*. Finally, Figure 15.2c shows the how these constraints are limited. In this case, the pronoun *she* can refer to *Abigail*, because the S non-terminal puts *Abigail* outside the domain of *she*. Similarly, *her* can also refer to *Abigail*. But *she* and *her* cannot be coreferent, because *she* c-commands *her*.

Heuristics

After applying constraints, heuristics are applied to select among the remaining candidates. Recency is a particularly strong heuristic. All things equal, readers will prefer the more recent referent for a given pronoun, particularly when comparing referents that occur in different sentences. Jurafsky and Martin (2009) offer the following example:

(15.3) The doctor found an old map in the captain's chest. Jim found an even older map hidden on the shelf. [It] described an island.

Readers are expected to prefer the older map as the referent for the pronoun *it*.

However, subjects are often preferred over objects, and this can contradict the preference for recency when two candidate referents are in the same sentence. For example,

(15.4) Abigail loaned Lucia a book on Spanish. [She] is always trying to help people.

Here, we may prefer to link *she* to *Abigail* rather than *Lucia*, because of *Abigail*'s position in the subject role of the preceding sentence. (Arguably, this preference would not be strong enough to select *Abigail* if the second sentence were *She is visiting Valencia next month*.)

A third heuristic is parallelism:

(15.5) Abigail loaned Lucia a book on Spanish. Özlem loaned [her] a book on Portuguese.

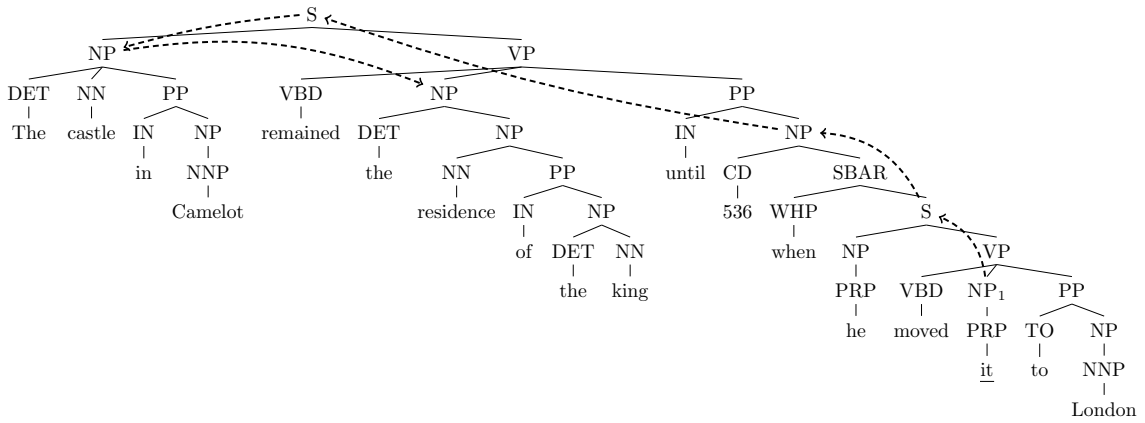

Figure 15.3: Left-to-right breadth-first tree traversal (Hobbs, 1978), indicating that the search for an antecedent for *it* (NP_1) would proceed in the following order: *536*; *the castle in Camelot*; *the residence of the king*; *Camelot*; *the king*. Hobbs (1978) proposes semantic constraints to eliminate *536* and *the castle in Camelot* as candidates, since they are unlikely to be the direct object of the verb *move*.

Here *Lucia* is preferred as the referent for *her*, contradicting the preference for the subject *Abigail* in the preceding example.

The recency and subject role heuristics can be unified by traversing the document in a syntax-driven fashion (Hobbs, 1978): each preceding sentence is traversed breadth-first, left-to-right (Figure 15.3). This heuristic successfully handles (15.4): *Abigail* is preferred as the referent for *she* because the subject NP is visited first. It also handles (15.3): the older map is preferred as the referent for *it* because the more recent sentence is visited first. (An alternative unification of recency and syntax is proposed by **centering theory** (Grosz et al., 1995), which is discussed in detail in chapter 16.)

In early work on reference resolution, the number of heuristics was small enough that a set of numerical weights could be set by hand (Lappin and Leass, 1994). More recent work uses machine learning to quantify the importance of each of these factors. However, pronoun resolution cannot be completely solved by constraints and heuristics alone. This is shown by the classic example pair (Winograd, 1972):

(15.6) The [city council] denied [the protesters] a permit because [they] advocated/feared violence.

Without reasoning about the motivations of the city council and protesters, it is unlikely that any system could correctly resolve both versions of this example.

Non-referential pronouns

While pronouns are generally used for reference, they need not refer to entities. The following examples show how pronouns can refer to propositions, events, and speech acts.

(15.7) a. They told me that I was too ugly for show business, but I didn't believe [it].
b. Elifsu saw Berthold get angry, and I saw [it] too.
c. Emmanuel said he worked in security. I suppose [that]'s one way to put it.

These forms of reference are generally not annotated in large-scale coreference resolution datasets such as OntoNotes (Pradhan et al., 2011).

Pronouns may also have **generic referents**:

(15.8) a. A poor carpenter blames [her] tools.
b. On the moon, [you] have to carry [your] own oxygen.
c. Every farmer who owns a donkey beats [it]. (Geach, 1962)

In the OntoNotes dataset, coreference is not annotated for generic referents, even in cases like these examples, in which the same generic entity is mentioned multiple times.

Some pronouns do not refer to anything at all:

(15.9) a. *[It]'s raining.*
[Il] pleut. (Fr)
b. [It] 's money that she's really after.
c. [It] is too bad that we have to work so hard.

How can we automatically distinguish these usages of *it* from referential pronouns? Consider the the difference between the following two examples (Bergsma et al., 2008):

(15.10) a. You can make [it] in advance.
b. You can make [it] in showbiz.

In the second example, the pronoun *it* is non-referential. One way to see this is by substituting another pronoun, like *them*, into these examples:

(15.11) a. You can make [them] in advance.
b. ? You can make [them] in showbiz.

The questionable grammaticality of the second example suggests that *it* is not referential. Bergsma et al. (2008) operationalize this idea by comparing distributional statistics for the n-grams around the word *it*, testing how often other pronouns or nouns appear in the same context. In cases where nouns and other pronouns are infrequent, the *it* is unlikely to be referential.

15.1.2 Proper Nouns

If a proper noun is used as a referring expression, it often corefers with another proper noun, so that the coreference problem is simply to determine whether the two names match. Subsequent proper noun references often use a shortened form, as in the running example (Figure 15.1):

(15.12) Apple Inc Chief Executive [Tim Cook] has jetted into China ... [Cook] is on his first business trip to the country ...

A typical solution for proper noun coreference is to match the syntactic head words of the reference with the referent. In § 10.5.2, we saw that the head word of a phrase can be identified by applying head percolation rules to the phrasal parse tree; alternatively, the head can be identified as the root of the dependency subtree covering the name. For sequences of proper nouns, the head word will be the final token.

There are a number of caveats to the practice of matching head words of proper nouns.

- In the European tradition, family names tend to be more specific than given names, and family names usually come last. However, other traditions have other practices: for example, in Chinese names, the family name typically comes first; in Japanese, honorifics come after the name, as in *Nobu-San* (*Mr. Nobu*).
- In organization names, the head word is often not the most informative, as in *Georgia Tech* and *Virginia Tech*. Similarly, *Lebanon* does not refer to the same entity as *Southern Lebanon*, necessitating special rules for the specific case of geographical modifiers (Lee et al., 2011).
- Proper nouns can be nested, as in *[the CEO of [Microsoft]]*, resulting in head word match without coreference.

Despite these difficulties, proper nouns are the easiest category of references to resolve (Stoyanov et al., 2009). In machine learning systems, one solution is to include a range of matching features, including exact match, head match, and string inclusion. In addition to matching features, competitive systems (e.g., Bengtson and Roth, 2008) include large lists, or **gazetteers**, of acronyms (e.g, *the National Basketball Association/NBA*), demonyms (e.g., *the Israelis/Israel*), and other aliases (e.g., *the Georgia Institute of Technology/Georgia Tech*).

15.1.3 Nominals

In coreference resolution, noun phrases that are neither pronouns nor proper nouns are referred to as **nominals**. In the running example (Figure 15.1), nominal references include: *the firm* (*Apple Inc*); *the firm's biggest growth market* (*China*); and *the country* (*China*).

Nominals are especially difficult to resolve (Denis and Baldridge, 2007; Durrett and Klein, 2013), and the examples above suggest why this may be the case: world knowledge is required to identify *Apple Inc* as a *firm*, and *China* as a *growth market*. Other difficult examples include the use of colloquial expressions, such as coreference between *Clinton campaign officials* and *the Clinton camp* (Soon et al., 2001).

15.2 Algorithms for coreference resolution

The ground truth training data for coreference resolution is a set of mention sets, where all mentions within each set refer to a single entity.[4] In the running example from Figure 15.1, the ground truth coreference annotation is:

$$\boldsymbol{c}_1 = \{\textit{Apple Inc}_{1:2}, \textit{the firm}_{27:28}\} \quad [15.1]$$
$$\boldsymbol{c}_2 = \{\textit{Apple Inc Chief Executive Tim Cook}_{1:6}, \textit{he}_{17}, \textit{Cook}_{33}, \textit{his}_{36}\} \quad [15.2]$$
$$\boldsymbol{c}_3 = \{\textit{China}_{10}, \textit{the firm 's biggest growth market}_{27:32}, \textit{the country}_{40:41}\} \quad [15.3]$$

Each row specifies the token spans that mention an entity. ("Singleton" entities, which are mentioned only once (e.g., *talks*, *government officials*), are excluded from the annotations.) Equivalently, if given a set of M mentions, $\{m_i\}_{i=1}^{M}$, each mention i can be assigned to a cluster z_i, where $z_i = z_j$ if i and j are coreferent. The cluster assignments z are invariant under permutation. The unique clustering associated with the assignment $\boldsymbol{z}$ is written $\boldsymbol{c}(\boldsymbol{z})$.

Coreference resolution can thus be viewed as a structure prediction problem, involving two subtasks: identifying which spans of text mention entities, and then clustering those spans.

Mention identification The task of identifying mention spans for coreference resolution is often performed by applying a set of heuristics to the phrase structure parse of each sentence. A typical approach is to start with all noun phrases and named entities, and then apply filtering rules to remove nested noun phrases with the same head (e.g., *[Apple CEO [Tim Cook]]*), numeric entities (e.g., *[100 miles]*, *[97%]*), non-referential *it*, etc (Lee et al., 2013; Durrett and Klein, 2013). In general, these deterministic approaches err in favor of recall, since the mention clustering component can choose to ignore false positive mentions, but cannot recover from false negatives. An alternative is to consider all spans (up to some finite length) as candidate mentions, performing mention identification and clustering jointly (Daumé III and Marcu, 2005; Lee et al., 2017).

[4] In many annotations, the term **markable** is used to refer to spans of text that can *potentially* mention an entity. The set of markables includes non-referential pronouns, which does not mention any entity. Part of the job of the coreference system is to avoid incorrectly linking these non-referential markables to any mention chains.

Mention clustering The subtask of mention clustering will be the focus of the remainder of this chapter. There are two main classes of models. In *mention-based models*, the scoring function for a coreference clustering decomposes over pairs of mentions. These pairwise decisions are then aggregated, using a clustering heuristic. Mention-based coreference clustering can be treated as a fairly direct application of supervised classification or ranking. However, the mention-pair locality assumption can result in incoherent clusters, like {*Hillary Clinton* ← *Clinton* ← *Mr Clinton*}, in which the pairwise links score well, but the overall result is unsatisfactory. *Entity-based models* address this issue by scoring entities holistically. This can make inference more difficult, since the number of possible entity groupings is exponential in the number of mentions.

15.2.1 Mention-pair models

In the **mention-pair model**, a binary label $y_{i,j} \in \{0, 1\}$ is assigned to each pair of mentions (i, j), where $i < j$. If i and j corefer ($z_i = z_j$), then $y_{i,j} = 1$; otherwise, $y_{i,j} = 0$. The mention *he* in Figure 15.1 is preceded by five other mentions: (1) *Apple Inc*; (2) *Apple Inc Chief Executive Tim Cook*; (3) *China*; (4) *talks*; (5) *government officials*. The correct mention pair labeling is $y_{2,6} = 1$ and $y_{i \neq 2,6} = 0$ for all other i. If a mention j introduces a new entity, such as mention 3 in the example, then $y_{i,j} = 0$ for all i. The same is true for "mentions" that do not refer to any entity, such as non-referential pronouns. If mention j refers to an entity that has been mentioned more than once, then $y_{i,j} = 1$ for all $i < j$ that mention the referent.

By transforming coreference into a set of binary labeling problems, the mention-pair model makes it possible to apply an off-the-shelf binary classifier (Soon et al., 2001). This classifier is applied to each mention j independently, searching backwards from j until finding an antecedent i which corefers with j with high confidence. After identifying a single **antecedent**, the remaining mention pair labels can be computed by transitivity: if $y_{i,j} = 1$ and $y_{j,k} = 1$, then $y_{i,k} = 1$.

Since the ground truth annotations give entity chains $\boldsymbol{c}$ but not individual mention-pair labels $\boldsymbol{y}$, an additional heuristic must be employed to convert the labeled data into training examples for classification. A typical approach is to generate at most one positive labeled instance $y_{a_j,j} = 1$ for mention j, where a_j is the index of the most recent antecedent, $a_j = \max\{i : i < j \wedge z_i = z_j\}$. Negative labeled instances are generated for all for all $i \in \{a_j + 1, \ldots, j\}$. In the running example, the most recent antecedent of the pronoun *he* is $a_6 = 2$, so the training data would be $y_{2,6} = 1$ and $y_{3,6} = y_{4,6} = y_{5,6} = 0$. The variable $y_{1,6}$ is not part of the training data, because the first mention appears before the true antecedent $a_6 = 2$.

15.2.2 Mention-ranking models

In **mention ranking** (Denis and Baldridge, 2007), the classifier learns to identify a single antecedent $a_i \in \{\epsilon, 1, 2, \ldots, i-1\}$ for each referring expression i,

$$\hat{a}_i = \underset{a \in \{\epsilon, 1, 2, \ldots, i-1\}}{\text{argmax}} \psi_M(a, i), \quad [15.4]$$

where $\psi_M(a, i)$ is a score for the mention pair (a, i). If $a = \epsilon$, then mention i does not refer to any previously-introduced entity — it is not **anaphoric**. Mention-ranking is similar to the mention-pair model, but all candidates are considered simultaneously, and at most a single antecedent is selected. The mention-ranking model explicitly accounts for the possibility that mention i is not anaphoric, through the score $\psi_M(\epsilon, i)$. The determination of anaphoricity can be made by a special classifier in a preprocessing step, so that non-ϵ antecedents are identified only for spans that are determined to be anaphoric (Denis and Baldridge, 2008).

As a learning problem, ranking can be trained using the same objectives as in discriminative classification. For each mention i, we can define a gold antecedent a_i^*, and an associated loss, such as the hinge loss, $\ell_i = (1 - \psi_M(a_i^*, i) + \psi_M(\hat{a}, i))_+$ or the negative log-likelihood, $\ell_i = -\log \text{p}(a_i^* \mid i; \boldsymbol{\theta})$. (For more on learning to rank, see § 17.1.1.) But as with the mention-pair model, there is a mismatch between the labeled data, which comes in the form of mention sets, and the desired supervision, which would indicate the specific antecedent of each mention. The antecedent variables $\{a_i\}_{i=1}^M$ relate to the mention sets in a many-to-one mapping: each set of antecedents induces a single clustering, but a clustering can correspond to many different settings of antecedent variables.

A heuristic solution is to set $a_i^* = \max\{j : j < i \wedge z_j = z_i\}$, the most recent mention in the same cluster as i. But the most recent mention may not be the most informative: in the running example, the most recent antecedent of the mention *Cook* is the pronoun *he*, but a more useful antecedent is the earlier mention *Apple Inc Chief Executive Tim Cook*. Rather than selecting a specific antecedent to train on, the antecedent can be treated as a latent variable, in the manner of the **latent variable perceptron** from § 12.4.2 (Fernandes et al., 2014):

$$\hat{\boldsymbol{a}} = \underset{\boldsymbol{a}}{\text{argmax}} \sum_{i=1}^{M} \psi_M(a_i, i) \quad [15.5]$$

$$\boldsymbol{a}^* = \underset{\boldsymbol{a} \in \mathcal{A}(\boldsymbol{c})}{\text{argmax}} \sum_{i=1}^{M} \psi_M(a_i, i) \quad [15.6]$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \sum_{i=1}^{M} \frac{\partial L}{\partial \boldsymbol{\theta}} \psi_M(a_i^*, i) - \sum_{i=1}^{M} \frac{\partial L}{\partial \boldsymbol{\theta}} \psi_M(\hat{a}_i, i) \quad [15.7]$$

where $\mathcal{A}(\boldsymbol{c})$ is the set of antecedent structures that is compatible with the ground truth coreference clustering $\boldsymbol{c}$. Another alternative is to sum over all the conditional probabilities of antecedent structures that are compatible with the ground truth clustering (Durrett and Klein, 2013; Lee et al., 2017). For the set of mention $\boldsymbol{m}$, we compute the following probabilities:

$$\mathrm{p}(\boldsymbol{c} \mid \boldsymbol{m}) = \sum_{\boldsymbol{a} \in \mathcal{A}(\boldsymbol{c})} \mathrm{p}(\boldsymbol{a} \mid \boldsymbol{m}) = \sum_{\boldsymbol{a} \in \mathcal{A}(\boldsymbol{c})} \prod_{i=1}^{M} \mathrm{p}(a_i \mid i, \boldsymbol{m}) \quad [15.8]$$

$$\mathrm{p}(a_i \mid i, \boldsymbol{m}) = \frac{\exp\left(\psi_M(a_i, i)\right)}{\sum_{a' \in \{\epsilon, 1, 2, \ldots, i-1\}} \exp\left(\psi_M(a', i)\right)}. \quad [15.9]$$

This objective rewards models that assign high scores for all valid antecedent structures. In the running example, this would correspond to summing the probabilities of the two valid antecedents for *Cook*, *he* and *Apple Inc Chief Executive Tim Cook*. In one of the exercises, you will compute the number of valid antecedent structures for a given clustering.

15.2.3 Transitive closure in mention-based models

A problem for mention-based models is that individual mention-level decisions may be incoherent. Consider the following mentions:

$$m_1 = \textit{Hillary Clinton} \quad [15.10]$$

$$m_2 = \textit{Clinton} \quad [15.11]$$

$$m_3 = \textit{Bill Clinton} \quad [15.12]$$

A mention-pair system might predict $\hat{y}_{1,2} = 1, \hat{y}_{2,3} = 1, \hat{y}_{1,3} = 0$. Similarly, a mention-ranking system might choose $\hat{a}_2 = 1$ and $\hat{a}_3 = 2$. Logically, if mentions 1 and 3 are both coreferent with mention 2, then all three mentions must refer to the same entity. This constraint is known as **transitive closure**.

Transitive closure can be applied *post hoc*, revising the independent mention-pair or mention-ranking decisions. However, there are many possible ways to enforce transitive closure: in the example above, we could set $\hat{y}_{1,3} = 1$, or $\hat{y}_{1,2} = 0$, or $\hat{y}_{2,3} = 0$. For documents with many mentions, there may be many violations of transitive closure, and many possible fixes. Transitive closure can be enforced by always adding edges, so that $\hat{y}_{1,3} = 1$ is preferred (e.g., Soon et al., 2001), but this can result in overclustering, with too many mentions grouped into too few entities.

Mention-pair coreference resolution can be viewed as a constrained optimization prob-

lem,

$$\max_{\boldsymbol{y} \in \{0,1\}^M} \quad \sum_{j=1}^{M} \sum_{i=1}^{j} \psi_M(i,j) \times y_{i,j}$$
$$\text{s.t.} \quad y_{i,j} + y_{j,k} - 1 \leq y_{i,k}, \quad \forall i < j < k,$$

with the constraint enforcing transitive closure. This constrained optimization problem is equivalent to graph partitioning with positive and negative edge weights: construct a graph where the nodes are mentions, and the edges are the pairwise scores $\psi_M(i,j)$; the goal is to partition the graph so as to maximize the sum of the edge weights between all nodes within the same partition (McCallum and Wellner, 2004). This problem is NP-hard, motivating approximations such as correlation clustering (Bansal et al., 2004) and **integer linear programming** (Klenner, 2007; Finkel and Manning, 2008, also see § 13.2.2).

15.2.4 Entity-based models

A weakness of mention-based models is that they treat coreference resolution as a classification or ranking problem, when it is really a clustering problem: the goal is to group the mentions together into clusters that correspond to the underlying entities. Entity-based approaches attempt to identify these clusters directly. Such methods require a scoring function at the entity level, measuring whether each set of mentions is internally consistent. Coreference resolution can then be viewed as the following optimization,

$$\max_{\boldsymbol{z}} \quad \sum_{e=1} \psi_E(\{i : z_i = e\}), \tag{15.13}$$

where z_i indicates the entity referenced by mention i, and $\psi_E(\{i : z_i = e\})$ is a scoring function applied to all mentions i that are assigned to entity e.

Entity-based coreference resolution is conceptually similar to the unsupervised clustering problems encountered in chapter 5: the goal is to obtain clusters of mentions that are internally coherent. The number of possible clusterings of n items is the **Bell number**, which is defined by the following recurrence (Bell, 1934; Luo et al., 2004),

$$B_n = \sum_{k=0}^{n-1} B_k \binom{n-1}{k} B_0 = \qquad\qquad B_1 = 1. \tag{15.14}$$

This recurrence is illustrated by the Bell tree, which is applied to a short coreference problem in Figure 15.4. The Bell number B_n grows exponentially with n, making exhaustive search of the space of clusterings impossible. For this reason, entity-based coreference resolution typically involves incremental search, in which clustering decisions are based on local evidence, in the hope of approximately optimizing the full objective in Equation 15.13. This approach is sometimes called **cluster ranking**, in contrast to mention ranking.

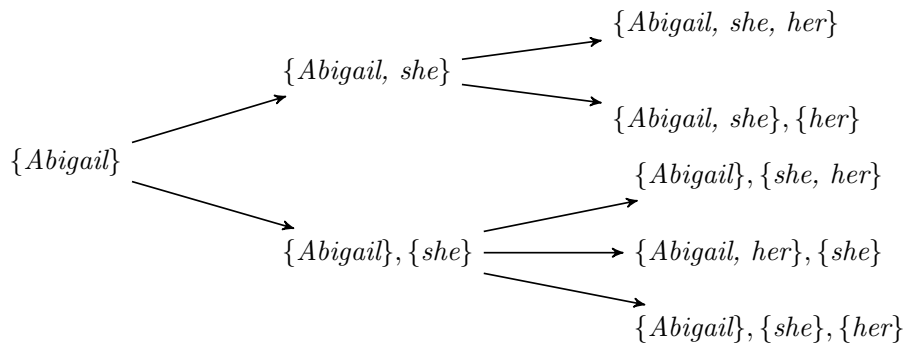

Figure 15.4: The Bell Tree for the sentence *Abigail hopes she speaks with her*. Which paths are excluded by the syntactic constraints mentioned in § 15.1.1?

***Generative models of coreference** Entity-based coreference can be approached through probabilistic **generative models**, in which the mentions in the document are conditioned on a set of latent entities (Haghighi and Klein, 2007, 2010). An advantage of these methods is that they can be learned from unlabeled data (Poon and Domingos, 2008, e.g.,); a disadvantage is that probabilistic inference is required not just for learning, but also for prediction. Furthermore, generative models require independence assumptions that are difficult to apply in coreference resolution, where the diverse and heterogeneous features do not admit an easy decomposition into mutually independent subsets.

Incremental cluster ranking

The SMASH method (§ 15.1.1) can be extended to entity-based coreference resolution by building up coreference clusters while moving through the document (Cardie and Wagstaff, 1999). At each mention, the algorithm iterates backwards through possible antecedent clusters; but unlike SMASH, a cluster is selected only if *all* members of its cluster are compatible with the current mention. As mentions are added to a cluster, so are their features (e.g., gender, number, animacy). In this way, incoherent chains like {*Hillary Clinton*, *Clinton*, *Bill Clinton*} can be avoided. However, an incorrect assignment early in the document — a **search error** — might lead to a cascade of errors later on.

More sophisticated search strategies can help to ameliorate the risk of search errors. One approach is **beam search** (first discussed in § 11.3), in which a set of hypotheses is maintained throughout search. Each hypothesis represents a path through the Bell tree (Figure 15.4). Hypotheses are "expanded" either by adding the next mention to an existing cluster, or by starting a new cluster. Each expansion receives a score, based on Equation 15.13, and the top K hypotheses are kept on the beam as the algorithm moves to the next step.

Incremental cluster ranking can be made more accurate by performing multiple passes over the document, applying rules (or "sieves") with increasing recall and decreasing precision at each pass (Lee et al., 2013). In the early passes, coreference links are proposed only between mentions that are highly likely to corefer (e.g., exact string match for full names and nominals). Information can then be shared among these mentions, so that when more permissive matching rules are applied later, agreement is preserved across the entire cluster. For example, in the case of $\{$*Hillary Clinton*, *Clinton*, *she*$\}$, the name-matching sieve would link *Clinton* and *Hillary Clinton*, and the pronoun-matching sieve would then link *she* to the combined cluster. A deterministic multi-pass system won nearly every track of the 2011 CoNLL shared task on coreference resolution (Pradhan et al., 2011). Given the dominance of machine learning in virtually all other areas of natural language processing — and more than fifteen years of prior work on machine learning for coreference — this was a surprising result, even if learning-based methods have subsequently regained the upper hand (e.g., Lee et al., 2018, the state of the art at the time of this writing).

Incremental perceptron

Incremental coreference resolution can be learned with the **incremental perceptron**, as described in § 11.3.2. At mention i, each hypothesis on the beam corresponds to a clustering of mentions $1 \dots i-1$, or equivalently, a path through the Bell tree up to position $i-1$. As soon as none of the hypotheses on the beam are compatible with the gold coreference clustering, a perceptron update is made (Daumé III and Marcu, 2005). For concreteness, consider a linear cluster ranking model,

$$\psi_E(\{i : z_i = e\}) = \sum_{i:z_i=e} \boldsymbol{\theta} \cdot \boldsymbol{f}(i, \{j : j < i \wedge z_j = e\}), \quad [15.15]$$

where the score for each cluster is computed as the sum of scores of all mentions that are linked into the cluster, and $\boldsymbol{f}(i, \varnothing)$ is a set of features for the non-anaphoric mention that initiates the cluster.

Using Figure 15.4 as an example, suppose that the ground truth is,

$$\boldsymbol{c}^* = \{\textit{Abigail, her}\}, \{\textit{she}\}, \quad [15.16]$$

but that with a beam of size one, the learner reaches the hypothesis,

$$\hat{\boldsymbol{c}} = \{\textit{Abigail, she}\}. \quad [15.17]$$

This hypothesis is incompatible with $\boldsymbol{c}^*$, so an update is needed:

$$\begin{aligned}
\boldsymbol{\theta} \leftarrow & \boldsymbol{\theta} + \boldsymbol{f}(\boldsymbol{c}^*) - \boldsymbol{f}(\hat{\boldsymbol{c}}) && [15.18]\\
= & \boldsymbol{\theta} + (\boldsymbol{f}(\textit{Abigail}, \varnothing) + \boldsymbol{f}(\textit{she}, \varnothing)) - (\boldsymbol{f}(\textit{Abigail}, \varnothing) + \boldsymbol{f}(\textit{she}, \{\textit{Abigail}\})) && [15.19]\\
= & \boldsymbol{\theta} + \boldsymbol{f}(\textit{she}, \varnothing) - \boldsymbol{f}(\textit{she}, \{\textit{Abigail}\}). && [15.20]
\end{aligned}$$

This style of incremental update can also be applied to a margin loss between the gold clustering and the top clustering on the beam. By backpropagating from this loss, it is also possible to train a more complicated scoring function, such as a neural network in which the score for each entity is a function of embeddings for the entity mentions (Wiseman et al., 2015).

Reinforcement learning

Reinforcement learning is a topic worthy of a textbook of its own (Sutton and Barto, 1998),[5] so this section will provide only a very brief overview, in the context of coreference resolution. A stochastic **policy** assigns a probability to each possible **action**, conditional on the context. The goal is to learn a policy that achieves a high expected reward, or equivalently, a low expected cost.

In incremental cluster ranking, a complete clustering on M mentions can be produced by a sequence of M actions, in which the action z_i either merges mention i with an existing cluster or begins a new cluster. We can therefore create a stochastic policy using the cluster scores (Clark and Manning, 2016),

$$\Pr(z_i = e; \boldsymbol{\theta}) = \frac{\exp \psi_E(i \cup \{j : z_j = e\}; \boldsymbol{\theta})}{\sum_{e'} \exp \psi_E(i \cup \{j : z_j = e'\}'; \boldsymbol{\theta})}, \quad [15.21]$$

where $\psi_E(i \cup \{j : z_j = e\}; \boldsymbol{\theta})$ is the score under parameters $\boldsymbol{\theta}$ for assigning mention i to cluster e. This score can be an arbitrary function of the mention i, the cluster e and its (possibly empty) set of mentions; it can also include the history of actions taken thus far.

If a policy assigns probability $\mathrm{p}(\boldsymbol{c}; \boldsymbol{\theta})$ to clustering $\boldsymbol{c}$, then its expected loss is,

$$L(\boldsymbol{\theta}) = \sum_{\boldsymbol{c} \in \mathcal{C}(\boldsymbol{m})} \mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{c}) \times \ell(\boldsymbol{c}), \quad [15.22]$$

where $\mathcal{C}(\boldsymbol{m})$ is the set of possible clusterings for mentions $\boldsymbol{m}$. The loss $\ell(\boldsymbol{c})$ can be based on any arbitrary scoring function, including the complex evaluation metrics used in coreference resolution (see § 15.4). This is an advantage of reinforcement learning, which can be trained directly on the evaluation metric — unlike traditional supervised learning, which requires a loss function that is differentiable and decomposable across individual decisions.

Rather than summing over the exponentially many possible clusterings, we can approximate the expectation by sampling trajectories of actions, $\boldsymbol{z} = (z_1, z_2, \ldots, z_M)$, from

[5] A draft of the second edition can be found here: `http://incompleteideas.net/book/the-book-2nd.html`. Reinforcement learning has been used in spoken dialogue systems (Walker, 2000) and text-based game playing (Branavan et al., 2009), and was applied to coreference resolution by Clark and Manning (2015).

the current policy. Each action z_i corresponds to a step in the Bell tree: adding mention m_i to an existing cluster, or forming a new cluster. Each trajectory $\boldsymbol{z}$ corresponds to a single clustering $\boldsymbol{c}$, and so we can write the loss of an action sequence as $\ell(\boldsymbol{c}(\boldsymbol{z}))$. The **policy gradient** algorithm computes the gradient of the expected loss as an expectation over trajectories (Sutton et al., 2000),

$$\frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{\theta}) = E_{\boldsymbol{z} \sim \mathcal{Z}(\boldsymbol{m})} \ell(\boldsymbol{c}(\boldsymbol{z})) \sum_{i=1}^{M} \frac{\partial}{\partial \boldsymbol{\theta}} \log \mathrm{p}(z_i \mid \boldsymbol{z}_{1:i-1}, \boldsymbol{m}) \quad [15.23]$$

$$\approx \frac{1}{K} \sum_{k=1}^{K} \ell(\boldsymbol{c}(\boldsymbol{z}^{(k)})) \sum_{i=1}^{M} \frac{\partial}{\partial \boldsymbol{\theta}} \log \mathrm{p}(z_i^{(k)} \mid \boldsymbol{z}_{1:i-1}^{(k)}, \boldsymbol{m}), \quad [15.24]$$

where each action sequence $\boldsymbol{z}^{(k)}$ is sampled from the current policy. Unlike the incremental perceptron, an update is not made until the complete action sequence is available.

Learning to search

Policy gradient can suffer from high variance: while the average loss over K samples is asymptotically equal to the expected reward of a given policy, this estimate may not be accurate unless K is very large. This can make it difficult to allocate credit and blame to individual actions. In **learning to search**, this problem is addressed through the addition of an **oracle** policy, which is known to receive zero or small loss. The oracle policy can be used in two ways:

- The oracle can be used to generate partial hypotheses that are likely to score well, by generating i actions from the initial state. These partial hypotheses are then used as starting points for the learned policy. This is known as **roll-in**.
- The oracle can be used to compute the minimum possible loss from a given state, by generating $M - i$ actions from the current state until completion. This is known as **roll-out**.

The oracle can be combined with the existing policy during both roll-in and roll-out, sampling actions from each policy (Daumé III et al., 2009). One approach is to gradually decrease the number of actions drawn from the oracle over the course of learning (Ross et al., 2011).

In the context of entity-based coreference resolution, Clark and Manning (2016) use the learned policy for roll-in and the oracle policy for roll-out. Algorithm 17 shows how the gradients on the policy weights are computed in this case. In this application, the oracle is "noisy", because it selects the action that minimizes only the *local* loss — the accuracy of the coreference clustering up to mention i — rather than identifying the action sequence that will lead to the best final coreference clustering on the entire document.

Algorithm 17 Learning to search for entity-based coreference resolution

procedure COMPUTE-GRADIENT(mentions $\boldsymbol{m}$, loss function ℓ, parameters $\boldsymbol{\theta}$)
  $L(\boldsymbol{\theta}) \leftarrow 0$
  $\boldsymbol{z} \sim \text{p}(\boldsymbol{z} \mid \boldsymbol{m}; \boldsymbol{\theta})$ ▷ Sample a trajectory from the current policy
  for $i \in \{1, 2, \ldots M\}$ **do**
    for action $z \in \mathcal{Z}(\boldsymbol{z}_{1:i-1}, \boldsymbol{m})$ **do** ▷ All possible actions after history $\boldsymbol{z}_{1:i-1}$
      $\boldsymbol{h} \leftarrow \boldsymbol{z}_{1:i-1} \oplus z$ ▷ Concatenate history $\boldsymbol{z}_{1:i-1}$ with action z
      for $j \in \{i+1, i+2, \ldots, M\}$ **do** ▷ Roll-out
        $h_j \leftarrow \text{argmin}_h \, \ell(\boldsymbol{h}_{1:j-1} \oplus h)$ ▷ Oracle selects action with minimum loss
      $L(\boldsymbol{\theta}) \leftarrow L(\boldsymbol{\theta}) + \text{p}(z \mid \boldsymbol{z}_{1:i-1}, \boldsymbol{m}; \boldsymbol{\theta}) \times \ell(\boldsymbol{h})$ ▷ Update expected loss
  return $\frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{\theta})$

When learning from noisy oracles, it can be helpful to mix in actions from the current policy with the oracle during roll-out (Chang et al., 2015).

15.3 Representations for coreference resolution

Historically, coreference resolution has employed an array of hand-engineered features to capture the linguistic constraints and preferences described in § 15.1 (Soon et al., 2001). Later work has documented the utility of lexical and bilexical features on mention pairs (Björkelund and Nugues, 2011; Durrett and Klein, 2013). The most recent and successful methods replace many (but not all) of these features with distributed representations of mentions and entities (Wiseman et al., 2015; Clark and Manning, 2016; Lee et al., 2017).

15.3.1 Features

Coreference features generally rely on a preprocessing pipeline to provide part-of-speech tags and phrase structure parses. This pipeline makes it possible to design features that capture many of the phenomena from § 15.1, and is also necessary for typical approaches to mention identification. However, the pipeline may introduce errors that propagate to the downstream coreference clustering system. Furthermore, the existence of such a pipeline presupposes resources such as treebanks, which do not exist for many languages.[6]

[6]The Universal Dependencies project has produced dependency treebanks for more than sixty languages. However, coreference features and mention detection are generally based on phrase structure trees, which exist for roughly two dozen languages. A list is available here: `https://en.wikipedia.org/wiki/Treebank`

Mention features

Features of individual mentions can help to predict anaphoricity. In systems where mention detection is performed jointly with coreference resolution, these features can also predict whether a span of text is likely to be a mention. For mention i, typical features include:

Mention type. Each span can be identified as a pronoun, name, or nominal, using the part-of-speech of the head word of the mention: both the Penn Treebank and Universal Dependencies tagsets (§ 8.1.1) include tags for pronouns and proper nouns, and all other heads can be marked as nominals (Haghighi and Klein, 2009).

Mention width. The number of tokens in a mention is a rough predictor of its anaphoricity, with longer mentions being less likely to refer back to previously-defined entities.

Lexical features. The first, last, and head words can help to predict anaphoricity; they are also useful in conjunction with features such as mention type and part-of-speech, providing a rough measure of agreement (Björkelund and Nugues, 2011). The number of lexical features can be very large, so it can be helpful to select only frequently-occurring features (Durrett and Klein, 2013).

Morphosyntactic features. These features include the part-of-speech, number, gender, and dependency ancestors.

The features for mention i and candidate antecedent a can be conjoined, producing joint features that can help to assess the compatibility of the two mentions. For example, Durrett and Klein (2013) conjoin each feature with the mention types of the anaphora and the antecedent. Coreference resolution corpora such as ACE and OntoNotes contain documents from various genres. By conjoining the genre with other features, it is possible to learn genre-specific feature weights.

Mention-pair features

For any pair of mentions i and j, typical features include:

Distance. The number of intervening tokens, mentions, and sentences between i and j can all be used as distance features. These distances can be computed on the surface text, or on a transformed representation reflecting the breadth-first tree traversal (Figure 15.3). Rather than using the distances directly, they are typically binned, creating binary features.

String match. A variety of string match features can be employed: exact match, suffix match, head match, and more complex matching rules that disregard irrelevant modifiers (Soon et al., 2001).

Compatibility. Building on the model, features can measure the anaphor and antecedent agree with respect to morphosyntactic attributes such as gender, number, and animacy.

Nesting. If one mention is nested inside another (e.g., *[The President of [France]]*), they generally cannot corefer.

Same speaker. For documents with quotations, such as news articles, personal pronouns can be resolved only by determining the speaker for each mention (Lee et al., 2013). Coreference is also more likely between mentions from the same speaker.

Gazetteers. These features indicate that the anaphor and candidate antecedent appear in a gazetteer of acronyms (e.g., *USA/United States*, *GATech/Georgia Tech*), demonyms (e.g., *Israel/Israeli*), or other aliases (e.g., *Knickerbockers/New York Knicks*).

Lexical semantics. These features use a lexical resource such as WORDNET to determine whether the head words of the mentions are related through synonymy, antonymy, and hypernymy (§ 4.2).

Dependency paths. The dependency path between the anaphor and candidate antecedent can help to determine whether the pair can corefer, under the government and binding constraints described in § 15.1.1.

Comprehensive lists of mention-pair features are offered by Bengtson and Roth (2008) and Rahman and Ng (2011). Neural network approaches use far fewer mention-pair features: for example, Lee et al. (2017) include only speaker, genre, distance, and mention width features.

Semantics In many cases, coreference seems to require knowledge and semantic inferences, as in the running example, where we link *China* with a *country* and a *growth market*. Some of this information can be gleaned from WORDNET, which defines a graph over **synsets** (see § 4.2). For example, one of the synsets of *China* is an instance of an `Asian_nation#1`, which in turn is a hyponym of `country#2`, a synset that includes *country*.[7] Such paths can be used to measure the similarity between concepts (Pedersen et al., 2004), and this similarity can be incorporated into coreference resolution as a feature (Ponzetto and Strube, 2006). Similar ideas can be applied to knowledge graphs induced from Wikipedia (Ponzetto and Strube, 2007). But while such approaches improve

[7] `teletype font` is used to indicate wordnet synsets, and *italics* is used to indicate strings.

relatively simple classification-based systems, they have proven less useful when added to the current generation of techniques.[8] For example, Durrett and Klein (2013) employ a range of semantics-based features — WordNet synonymy and hypernymy relations on head words, named entity types (e.g., person, organization), and unsupervised clustering over nominal heads — but find that these features give minimal improvement over a baseline system using surface features.

Entity features

Many of the features for entity-mention coreference are generated by aggregating mention-pair features over all mentions in the candidate entity (Culotta et al., 2007; Rahman and Ng, 2011). Specifically, for each binary mention-pair feature $f(i, j)$, we compute the following entity-mention features for mention i and entity $e = \{j : j < i \wedge z_j = e\}$.

- ALL-TRUE: Feature $f(i, j)$ holds for all mentions $j \in e$.
- MOST-TRUE: Feature $f(i, j)$ holds for at least half and fewer than all mentions $j \in e$.
- MOST-FALSE: Feature $f(i, j)$ holds for at least one and fewer than half of all mentions $j \in e$.
- NONE: Feature $f(i, j)$ does not hold for any mention $j \in e$.

For scalar mention-pair features (e.g., distance features), aggregation can be performed by computing the minimum, maximum, and median values across all mentions in the cluster. Additional entity-mention features include the number of mentions currently clustered in the entity, and ALL-X and MOST-X features for each mention type.

15.3.2 Distributed representations of mentions and entities

Recent work has emphasized distributed representations of both mentions and entities. One potential advantage is that pre-trained embeddings could help to capture the semantic compatibility underlying nominal coreference, helping with difficult cases like (*Apple*, *the firm*) and (*China*, *the firm's biggest growth market*). Furthermore, a distributed representation of entities can be trained to capture semantic features that are added by each mention.

Mention embeddings

Entity mentions can be embedded into a vector space, providing the base layer for neural networks that score coreference decisions (Wiseman et al., 2015).

[8] This point was made by Michael Strube at a 2015 workshop, noting that as the quality of the machine learning models in coreference has improved, the benefit of including semantics has become negligible.

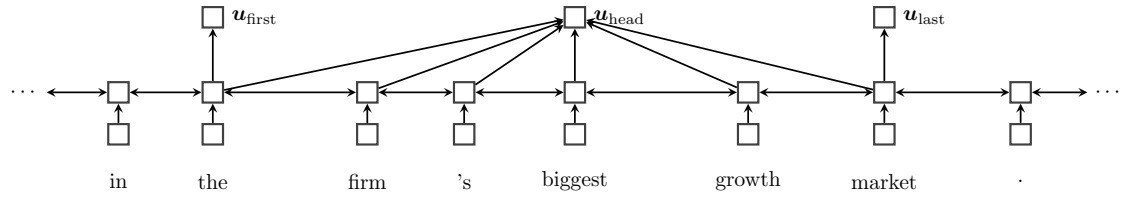

Figure 15.5: A bidirectional recurrent model of mention embeddings. The mention is represented by its first word, its last word, and an estimate of its head word, which is computed from a weighted average (Lee et al., 2017).

Constructing the mention embedding Various approaches for embedding multiword units can be applied (see § 14.8). Figure 15.5 shows a recurrent neural network approach, which begins by running a bidirectional LSTM over the entire text, obtaining hidden states from the left-to-right and right-to-left passes, $\boldsymbol{h}_m = [\overleftarrow{\boldsymbol{h}}_m; \overrightarrow{\boldsymbol{h}}_m]$. Each candidate mention span (s, t) is then represented by the vertical concatenation of four vectors:

$$\boldsymbol{u}^{(s,t)} = [\boldsymbol{u}^{(s,t)}_{\text{first}}; \boldsymbol{u}^{(s,t)}_{\text{last}}; \boldsymbol{u}^{(s,t)}_{\text{head}}; \phi^{(s,t)}], \tag{15.25}$$

where $\boldsymbol{u}^{(s,t)}_{\text{first}} = \boldsymbol{h}_{s+1}$ is the embedding of the first word in the span, $\boldsymbol{u}^{(s,t)}_{\text{last}} = \boldsymbol{h}_t$ is the embedding of the last word, $\boldsymbol{u}^{(s,t)}_{\text{head}}$ is the embedding of the "head" word, and $\phi^{(s,t)}$ is a vector of surface features, such as the length of the span (Lee et al., 2017).

Attention over head words Rather than identifying the head word from the output of a parser, it can be computed from a neural **attention mechanism**:

$$\tilde{\alpha}_m = \boldsymbol{\theta}_\alpha \cdot \boldsymbol{h}_m \tag{15.26}$$

$$\boldsymbol{a}^{(s,t)} = \text{SoftMax}\left([\tilde{\alpha}_{s+1}, \tilde{\alpha}_{s+2}, \ldots, \tilde{\alpha}_t]\right) \tag{15.27}$$

$$\boldsymbol{u}^{(s,t)}_{\text{head}} = \sum_{m=s+1}^{t} a^{(s,t)}_m \boldsymbol{h}_m. \tag{15.28}$$

Each token m gets a scalar score $\tilde{\alpha}_m = \boldsymbol{\theta}_\alpha \cdot \boldsymbol{h}_m$, which is the dot product of the LSTM hidden state $\boldsymbol{h}_m$ and a vector of weights $\boldsymbol{\theta}_\alpha$. The vector of scores for tokens in the span $m \in \{s+1, s+2, \ldots, t\}$ is then passed through a softmax layer, yielding a vector $\boldsymbol{a}^{(s,t)}$ that allocates one unit of attention across the span. This eliminates the need for syntactic parsing to recover the head word; instead, the model learns to identify the most important words in each span. Attention mechanisms were introduced in neural machine translation (Bahdanau et al., 2014), and are described in more detail in § 18.3.1.

Using mention embeddings Given a set of mention embeddings, each mention i and candidate antecedent a is scored as,

$$\psi(a, i) = \psi_S(a) + \psi_S(i) + \psi_M(a, i) \quad [15.29]$$
$$\psi_S(a) = \text{FeedForward}_S(\boldsymbol{u}^{(a)}) \quad [15.30]$$
$$\psi_S(i) = \text{FeedForward}_S(\boldsymbol{u}^{(i)}) \quad [15.31]$$
$$\psi_M(a, i) = \text{FeedForward}_M([\boldsymbol{u}^{(a)}; \boldsymbol{u}^{(i)}; \boldsymbol{u}^{(a)} \odot \boldsymbol{u}^{(i)}; \boldsymbol{f}(a, i, \boldsymbol{w})]), \quad [15.32]$$

where $\boldsymbol{u}^{(a)}$ and $\boldsymbol{u}^{(i)}$ are the embeddings for spans a and i respectively, as defined in Equation 15.25.

- The scores $\psi_S(a)$ quantify whether span a is likely to be a coreferring mention, independent of what it corefers with. This allows the model to learn identify mentions directly, rather than identifying mentions with a preprocessing step.
- The score $\psi_M(a, i)$ computes the compatibility of spans a and i. Its base layer is a vector that includes the embeddings of spans a and i, their elementwise product $\boldsymbol{u}^{(a)} \odot \boldsymbol{u}^{(i)}$, and a vector of surface features $\boldsymbol{f}(a, i, \boldsymbol{w})$, including distance, speaker, and genre information.

Lee et al. (2017) provide an error analysis that shows how this method can correctly link a *blaze* and a *fire*, while incorrectly linking *pilots* and *fight attendants*. In each case, the coreference decision is based on similarities in the word embeddings.

Rather than embedding individual mentions, Clark and Manning (2016) embed mention pairs. At the base layer, their network takes embeddings of the words in and around each mention, as well as one-hot vectors representing a few surface features, such as the distance and string matching features. This base layer is then passed through a multilayer feedforward network with ReLU nonlinearities, resulting in a representation of the mention pair. The output of the mention pair encoder $\boldsymbol{u}_{i,j}$ is used in the scoring function of a mention-ranking model, $\psi_M(i, j) = \boldsymbol{\theta} \cdot \boldsymbol{u}_{i,j}$. A similar approach is used to score cluster pairs, constructing a cluster-pair encoding by **pooling** over the mention-pair encodings for all pairs of mentions within the two clusters.

Entity embeddings

In entity-based coreference resolution, each entity should be represented by properties of its mentions. In a distributed setting, we maintain a set of vector entity embeddings, $\boldsymbol{v}_e$. Each candidate mention receives an embedding $\boldsymbol{u}_i$; Wiseman et al. (2016) compute this embedding by a single-layer neural network, applied to a vector of surface features. The decision of whether to merge mention i with entity e can then be driven by a feedforward

network, $\psi_E(i, e) = \text{Feedforward}([\boldsymbol{v}_e; \boldsymbol{u}_i])$. If i is added to entity e, then its representation is updated recurrently, $\boldsymbol{v}_e \leftarrow f(\boldsymbol{v}_e, \boldsymbol{u}_i)$, using a recurrent neural network such as a long short-term memory (LSTM; chapter 6). Alternatively, we can apply a pooling operation, such as max-pooling or average-pooling (chapter 3), setting $\boldsymbol{v}_e \leftarrow \text{Pool}(\boldsymbol{v}_e, \boldsymbol{u}_i)$. In either case, the update to the representation of entity e can be thought of as adding new information about the entity from mention i.

15.4 Evaluating coreference resolution

The state of coreference evaluation is aggravatingly complex. Early attempts at simple evaluation metrics were found to be susceptible to trivial baselines, such as placing each mention in its own cluster, or grouping all mentions into a single cluster. Following Denis and Baldridge (2009), the CoNLL 2011 shared task on coreference (Pradhan et al., 2011) formalized the practice of averaging across three different metrics: MUC (Vilain et al., 1995), B-CUBED (Bagga and Baldwin, 1998a), and CEAF (Luo, 2005). Reference implementations of these metrics are available from Pradhan et al. (2014) at `https://github.com/conll/reference-coreference-scorers`.

Additional resources

Ng (2010) surveys coreference resolution through 2010. Early work focused exclusively on pronoun resolution, with rule-based (Lappin and Leass, 1994) and probabilistic methods (Ge et al., 1998). The full coreference resolution problem was popularized in a shared task associated with the sixth Message Understanding Conference, which included coreference annotations for training and test sets of thirty documents each (Grishman and Sundheim, 1996). An influential early paper was the decision tree approach of Soon et al. (2001), who introduced mention ranking. A comprehensive list of surface features for coreference resolution is offered by Bengtson and Roth (2008). Durrett and Klein (2013) improved on prior work by introducing a large lexicalized feature set; subsequent work has emphasized neural representations of entities and mentions (Wiseman et al., 2015).

Exercises

1. Select an article from today's news, and annotate coreference for the first twenty noun phrases and possessive pronouns that appear in the article, include ones that are nested within larger noun phrases. Then specify the mention-pair training data that would result from the first five of these candidate entity mentions.

2. Using your annotations from the preceding problem, compute the following statistics:

- The number of times new entities are introduced by each of the three types of referring expressions: pronouns, proper nouns, and nominals. Include "singleton" entities that are mentioned only once.
- For each type of referring expression, compute the fraction of mentions that are anaphoric.

3. Apply a simple heuristic to all pronouns in the article from the previous exercise: link each pronoun to the closest preceding noun phrase that agrees in gender, number, animacy, and person. Compute the following evaluation:

   - True positive: a pronoun that is linked to a noun phrase with which it is coreferent, or is labeled as the first mention of an entity when in fact it does not corefer with any preceding mention. In this case, non-referential pronouns can be true positives if they are marked as having no antecedent.
   - False positive: a pronoun that is linked to a noun phrase with which it is not coreferent. This includes mistakenly linking singleton or non-referential pronouns.
   - False negative: a pronoun that has at least one antecedent, but is either labeled as not having an antecednet, or is linked to mention with which it does not corefer.

   Compute the F-MEASURE for your method, and for a trivial baseline in which every pronoun refers to the immediately preceding entity mention. Are there any additional heuristics that would have improved the performance of this method?

4. Durrett and Klein (2013) compute the probability of the gold coreference clustering by summing over all antecedent structures that are compatible with the clustering. For example, if there are three mentions of a single entity, m_1, m_2, m_3, there are two possible antecedent structures: $a_2 = 1, a_3 = 1$ and $a_2 = 1, a_3 = 2$. Compute the number of antecedent structures for a single entity with K mentions.

5. Suppose that all mentions can be unambiguously divided into C classes, for example by gender and number. Further suppose that mentions from different classes can never corefer. In a document with M mentions, give upper and lower bounds on the total number of possible coreference clusterings, in terms of the Bell numbers and the parameters M and C. Compute numerical upper and lower bounds for the case $M = 4, C = 2$.

6. Lee et al. (2017) propose a model that considers all contiguous spans in a document as possible mentions.

   a) In a document of length M, how many mention pairs must be evaluated? (All answers can be given in asymptotic, big-O notation.)

b) To make inference more efficient, Lee et al. (2017) restrict consideration to spans of maximum length $L \ll M$. Under this restriction, how many mention pairs must be evaluated?

c) To further improve inference, one might evaluate coreference only between pairs of mentions whose endpoints are separated by a maximum of D tokens. Under this additional restriction, how many mention pairs must be evaluated?

7. In Spanish, the subject can be omitted when it is clear from context, e.g.,

(15.13) *Las ballenas no son peces. Son mamíferos.*
The whales no are fish. Are mammals.
Whales are not fish. They are mammals.

Resolution of such **null subjects** is facilitated by the Spanish system of verb morphology, which includes distinctive suffixes for most combinations of person and number. For example, the verb form *son* ('are') agrees with the third-person plural pronouns *ellos* (masculine) and *ellas* (feminine), as well as the second-person plural *ustedes*.

Suppose that you are given the following components:

- A system that automatically identifies verbs with null subjects.
- A function $c(j, p) \in \{0, 1\}$ that indicates whether pronoun p is compatible with null subject j, according to the verb morphology.
- A trained mention-pair model, which computes scores $\psi(w_i, w_j, j - i) \in \mathbb{R}$ for all pairs of mentions i and j, scoring the pair by the antecedent mention w_i, the anaphor w_j, and the distance $j - i$.

Describe an integer linear program that simultaneously performs two tasks: resolving coreference among all entity mentions, and identifying suitable pronouns for all null subjects. In the example above, your program should link the null subject with *las ballenas* ('whales'), and identify *ellas* as the correct pronoun. For simplicity, you may assume that null subjects cannot be antecedents, and you need not worry about the transitivity constraint described in § 15.2.3.

8. Use the policy gradient algorithm to compute the gradient for the following scenario, based on the Bell tree in Figure 15.4:

- The gold clustering $\boldsymbol{c}^*$ is $\{\textit{Abigail}, \textit{her}\}, \{\textit{she}\}$.

- Drawing a single sequence of actions ($K = 1$) from the current policy, you obtain the following incremental clusterings:

$$\begin{aligned}\boldsymbol{c}(\boldsymbol{a}_1) &= \{\textit{Abigail}\} \\ \boldsymbol{c}(\boldsymbol{a}_{1:2}) &= \{\textit{Abigail, she}\} \\ \boldsymbol{c}(\boldsymbol{a}_{1:3}) &= \{\textit{Abigail, she}\}, \{\textit{her}\}.\end{aligned}$$

- At each mention t, the space of actions $\mathcal{A}_t$ includes merging the mention with each existing cluster or with the empty cluster. The probability of merging m_t with cluster c is proportional to the exponentiated score for the merged cluster,

$$\text{p}(\text{Merge}(m_t, c))) \propto \exp \psi_E(m_t \cup c), \quad [15.33]$$

  where $\psi_E(m_t \cup c)$ is defined in Equation 15.15.

Compute the gradient $\frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{\theta})$ in terms of the loss $\ell(\boldsymbol{c}(\boldsymbol{a}))$ and the features of each (potential) cluster. Explain the differences between the gradient-based update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{\theta})$ and the incremental perceptron update from this same example.

9. As discussed in § 15.1.1, some pronouns are not referential. In English, this occurs frequently with the word *it*. Download the text of *Alice in Wonderland* from NLTK, and examine the first ten appearances of *it*. For each occurrence:

   - First, examine a five-token window around the word. In the first example, this window is,

     `, but it had no`

     Is there another pronoun that could be substituted for *it*? Consider *she*, *they*, and *them*. In this case, both *she* and *they* yield grammatical substitutions. What about the other ten appearances of *it*?
   - Now, view an fifteen-word window for each example. Based on this window, mark whether you think the word *it* is referential.

   How often does the substitution test predict whether *it* is referential?

10. Now try to automate the test, using the Google n-grams corpus (Brants and Franz, 2006). Specifically, find the count of each 5-gram containing *it*, and then compute the counts of 5-grams in which *it* is replaced with other third-person pronouns: *he*, *she*, *they*, *her*, *him*, *them*, *herself*, *himself*.

    There are various ways to get these counts. One approach is to download the raw data and search it; another is to construct web queries to `https://books.google.com/ngrams`.

Compare the ratio of the counts of the original 5-gram to the summed counts of the 5-grams created by substitution. Is this ratio a good predictor of whether *it* is referential?