Text Simplification for Information-Seeking Applications

Beata Beigman Klebanov[1], Kevin Knight[2], and Daniel Marcu[2]

[1] The Hebrew University, Jerusalem, 91904, Israel
beata@cs.huji.ac.il,
http://www.cs.huji.ac.il/~beata
[2] Information Science Institute, University of Southern California
90292 Marina Del Rey, CA, USA
{knight,marcu}@isi.edu
http://www.isi.edu/{~marcu, ~knight}

Abstract. This paper addresses the issue of simplifying natural language texts in order to ease the task of accessing factual information contained in them. We define the notion of Easy Access Sentence - a unit of text from which the information it contains can be retrieved by a system with modest text-analysis capabilities, able to process single verb sentences with named entities as constituents. We present an algorithm that constructs Easy Access Sentences from the input text, with a small-scale evaluation. Challenges and further research directions are then discussed.

1 Introduction

It has been argued previously that complicated sentences are a stumbling block for systems that rely on natural language data; applications like machine translation, information retrieval and text summarization were cited as potential benefactors of text simplification [5][6]. However, what exactly makes a sentence simple for computers has not yet been made clear.

Possible dimensions of complexity are numerous. Long sentences, conjoined sentences, embedded clauses, passives, non-canonical word order [4], use of low-frequency words [7] were all proposed as aspects of sentence complexity for language-impaired humans. Are the same things difficult for computers? Why?

The crucial question is what language technology applications use texts for. Taggers and parsers of various sorts perform ***linguistic analysis of the text*** and are hence pre-processors for applications that make use of the (analyzed) text, usually for ***finding information in it***. This goal statement pertains information retrieval and extraction, to question answering and summarization. Machine translation systems might also have an information-seeking component if translation is viewed as a task of conveying the same message in a different language, rather than transforming the structures of one language to those of the other.

R. Meersman, Z. Tari (Eds.): CoopIS/DOA/ODBASE 2004, LNCS 3290, pp. 735–747, 2004.

In this paper, we address the question of what makes finding information in a text easy for a computer and how to transform texts to comply with these requirements.

2 Easy Access Sentences

Intuitively, a simple sentence is a sentence from which it is easy to retrieve the information it contains. For example, consider the following sentences that all convey the fact that Bill Clinton married Hillary Rodham in 1975.

1. Bill Clinton married Hillary Rodham in 1975.
2. Bill Clinton graduated from Yale in 1973 and married Hillary Rodham in 1975.
3. After marrying Hillary Rodham in 1975, Bill Clinton started a career as a politician.
4. Bill Clinton met Hillary Rodham in Yale, and married her in 1975.
5. Bill Clinton met Hillary Rodham in the early 1970s; their wedding took place in 1975.
6. Bill Clinton was introduced to the Rodhams in the early 1970s, and married their daughter Hillary in 1975.

Consider the processes involved in retrieving the information "Bill Clinton married Hillary Rodham in 1975". Example (1) states just this in a concise and explicit fashion. To get the information from (2), one needs to retrieve the subject of the verb "married" from elsewhere in the sentence; (3) requires in addition assigning tense to "marrying"; (4) needs subject retrieval and resolution of the anaphor "her" to Hillary Clinton. To handle example (5), the system should also possess some lexical knowledge (having a wedding is equivalent to getting married); example (6) assumes world-knowledge based inference (a daughter's family name is usually the same as her parents'). While the exact degree of difficulty depends on the information-seeking system's having the appropriate knowledge sources and skills, (1) is clearly the least demanding case. Our model example being (1), we define:

Easy Access Sentence. *EAS* based on a text *T* satisfies the following requirements:

Sentence. *EAS* is a grammatical sentence;

Single Verb. *EAS* has one finite[1] verb;

Information Maintenance. *EAS* does not make any claims that were not present, explicitly or implicitly, in *T*;

Named Entities. The more Named Entities a sentence satisfying the previous three requirements contains, the better *EAS* it is.

The first requirement ensures that sub-sentential entities are excluded; thus, *married Hillary Clinton* is not an EAS.

[1] A finite verb is a verb in some tense - present, past, future.

The Single Verb requirement eliminates the need to assign tense to the verb (*Bill Clinton marrying Hillary Clinton* is not an EAS) and to retrieve a dependent[2] of a verb from the dependency structure of another verb.

Information Maintenance ensures that when representing a text as a set of EASes based on it, we do not introduce information that was not in the text. For example, if an information-seeking system resolves *her* in example (4) above to Yale, it could produce a putative EAS *Bill Clinton married Yale in 1975*, which would fail the Information Maintenance requirement.

The drive towards Named Entities encodes preference of sentences with full names of entities to sentences with partial or indirect references to the entities which need to be resolved, like pronouns (*he, she*), partial names (*Mr Clinton*), definite noun phrases (*the former president of the United States*).

3 Text Based EASes

To exemplify the notion of EASes based on a text, let us consider a stretch of text converted by hand into a set of Easy Access Sentences. The example is adapted from a biography of Harriet Beecher Stowe:

> Harriet Beecher Stowe is a writer. She was born in Litchfield, Connecticut, USA, the daughter of Lyman Beecher. Raised by her severe Calvinist father, she was educated and then taught at the Hartford Female Seminary (founded by her sister Catherine Beecher). Moving to Cincinnati with her father (1832), she began to write short fiction, and after her marriage (1836) persevered in her writing while raising seven children.

Had we been able to rewrite the text into the following set of sentences, applications that are looking for information about this 19th century writer would have found it easily.

- Harriet Beecher Stowe is a writer.
- Harriet Beecher Stowe was born in Litchfield, Connecticut, USA.
- Harriet Beecher Stowe is the daughter of Lyman Beecher.
- Harriet Beecher Stowe was raised by her severe Calvinist father.
- Harriet Beecher Stowe was raised by Lyman Beecher.
- Lyman Beecher is Harriet Beecher Stowe's father.
- Harriet Beecher Stowe was educated at the Hartford Female Seminary.
- Harriet Beecher Stowe taught at the Hartford Female Seminary.
- Catherine Beecher founded the Hartford Female Seminary.
- Catherine Beecher is Harriet Beecher Stowe's sister.
- Harriet Beecher Stowe moved to Cincinnati with her father in 1832.
- Harriet Beecher Stowe moved to Cincinnati with Lyman Beecher in 1832.
- Harriet Beecher Stowe wrote short fiction.
- Harriet Beecher Stowe married in 1836.
- Harriet Beecher Stowe raised seven children.

[2] Verb dependents are subject, direct and indirect objects, modifiers.

All of the above sentences comply with the EAS requirements - each is a grammatical sentence with one tensed verb reporting a piece of information explicitly or implicitly present in the original text (for example, the fact that Lyman Beecher is Harriet Beecher Stowe's father is not stated explicitly, but is a correct inference from the text). Pronouns and some other anaphoric elements (like *her severe Calvinist father*) are substituted with the appropriate names. Now pieces of factual information about Harriet Beecher Stowe, like date of marriage, father's name, birth place, number of children can all be retrieved using relatively simple tools.

Our aim is automatic construction of EASes from a text. It can be argued that if it is possible to construct them automatically, this could as well be done by the information-seeking application itself, using the very same tools and methods we will be using.

We note that information-seeking applications are usually quite complex systems that have to worry about many things other than those involved in EAS-construction, like query formulation, search algorithm and validation of the answer (in question answering and information retrieval), lexicon translation and text generation in another language (for a machine translation system), database maintenance and employment (for applications that mine data for future use). Thus, it would be useful to outsource a part of text analysis to a specially designed mechanism that produces a representation from which the information contained in the text can be easily accessed.

In addition, many state-of-the-art language processing systems [9] operate on phrase or word level; hence information scattered across a number of phrases or even sentences is difficult to pinpoint and consolidate. Information dispersion, however, is quite abundant; the resolution of an anaphor can be a number of sentences back; the correct tense of the verb needs to be inferred by looking at the governing verb and possibly other things; the implicit subject of a verb in a relative clause resides somewhere in the area of the main clause of the sentence. Thus, bringing related pieces of information closer together and structuring them in a certain pre-defined way might help increase the accuracy and coverage of these systems.

Finally, as sentences containing a single verb and its dependents, EASes lend themselves to coding into databases that can later be re-used as external knowledge sources for various applications.

4 Constructing EASes

In this section, we present an algorithm for constructing EASes from a given text, and discuss our implementation of the key issues.

4.1 Main Algorithm

We first identify the person names in a text using BBN's Identifinder [2] and derive dependency structures for its sentences using MINIPAR [8]. We then

proceed verb-wise, trying to construct an EAS with this verb as its single finite verb. Hence, for every verb *V*:

1. Check if an EAS with *V* is in a semantically problematic environment (see section 4.2 for details). If it is, skip *V* and proceed to the next verb.
2. If *V* is not finite, assign tense (section 4.3).
3. Collect *V*'s dependents *Deps* (section 4.4).
4. Try to increase the number of Named Entities among *Deps* (section 4.5).
5. Output an EAS containing *V* and *Deps*.

Appositions are treated as if they were dependents of the verb *is*. Hence, an apposition like *George Bush, the president of the US...* is turned into *George Bush is the president of the US.* We use MINIPAR to detect appositions, and currently process only those that mention a person name.

MINIPAR's output eliminates lexical realizations of conjunctions; hence there is no way to differentiate between *and* and *or*. When outputting EASes, we substitute *and* for every conjunction node. While this is an error-prone procedure (for example, *The benchmark tumbled 301.24 points, or 1.06 percent* turns into *The benchmark tumbled 301.24 points and 1.06 percent*), we have not yet implemented a device to track down the original lexical realization of the conjunction.

4.2 Semantically Problematic Environments

Certain constructions do not contain factual information, and thus are not amenable to transformation into EAS. Consider:

- If Jane *arrives* early, John will be happy.
- I did not see John *coming*.
- George believes that Helen *died* yesterday.

For all the italicized verbs, there is no simple tense we can put them into such that an EAS centered around them would pass Information Maintenance test: none of *Jane arrives early*, *Jane arrived early*, *Jane will arrive early* represents information contained in the original sentence. Similarly, we can't derive any definite statement about John's coming or Helen's death.

One can envision an implementation where both *Jane will possibly leave early* and *Jane will possibly not leave early* are generated; however, the value of these EASes for information-seeking applications is doubtful. The current implementation uses lists of conditional markers, negation, verbs not presupposing their sentential, gerundive and infinitival complements[3] to detect governors[4] of these kinds, and avoids extraction of EASes from their domains.

Modality is another semantically problematic environment. Although *IBM started laying off employees* means that *IBM lays off employees*, once modality is applied, the inference does not hold anymore. Hence, *IBM*

[3] We used lexical units from *attempt, cogitation, desiring, request* and other frames of FrameNet [1] to help construct these verb lists.

[4] A governor of a node *N* is a node within the transitive closure of the is-a-dependent-of relation, starting from *N*.

might/should/would/must start laying off employees does not yield *IBM lays off employees*. The current implementation does not build any EASes from sentences with modals; further research is needed to see whether a definite negative statement can be produced: *IBM might start laying off people* means that *IBN does not lay off people*.

Checking 123 putative EASes generated by our system from 10 subsequent newswire articles from a random TREC-2002 [11] document (henceforth TestSet), we found 5 cases of erroneous extraction from a semantically problematic environment. 4 were due to the non-presupposing governor missing from our list; 3 were due to parser errors where the governor was mis-identified[5].

4.3 Tense Assignment

To assign tense to an infinitival or a gerundive verb, we go up the dependency structure and assign the tense of the closest tensed governor. Hence, *Jane continued writing* would yield *Jane wrote*. In the TestSet, there were 26 cases of tense assignment, out of which 17 were correct (65.4%).

Wrong tense assignment means a mistake in building the tense (ex. *helded* as past tense of *held*), or non-compliance with Information Maintenance. As an example of the latter, consider inferring *The squad prepared for next year's internationals* and *Next year's internationals included the World Cup* from the following sentence: *John Hart named a 42-man squad to prepare for next year's internationals, including the World Cup*. In both cases the past tense was taken from that of *named*, instead of the correct present tense.

4.4 Collecting Verb's Dependents

The dependency information is given in the output of MINIPAR. We recursively collect dependents of the verb ignoring verb-level conjunctions, relative clauses[6] and the surface subject (marked *s*).

When the deep subject of the verb (marked *subj*) is an empty string, we follow the antecedence links provided by MINIPAR to retrieve the subject. If this does not help, we default to the subject of the clause to which the current clause is attached[7]. In the TestSet, 33 cases needed subject retrieval, 17 of which were treated correctly. 13 mistakes were due to MINIPAR's incorrect antecedence links, 1 - to a mistake in the dependency structure returned by MINIPAR, 1 - to our procedure of substituting *and* for conjunctions, and in one case our default subject retrieval algorithm produced an incorrect result.

Out of the 123 sentences in the TestSet, 26 contained mistakes in verb's dependents other than the subject. 18 of those were due to MINIPAR's misparsing the clause, 7 were due to the conjunction substitution procedure and

[5] In 2 of the 5 cases both failures happened - the parser mis-identified the non-presupposing governor, but even had it been identified correctly, the EAS-construction software would have erred, since this governor was missing from the list.

[6] to comply with the Single Verb requirement

[7] See Clinton-Rodham examples 2 and 3 in section 2.

one was due to dropping the relative clause which turned out to be a restrictive one, hence the resulting general meaning was not supported: we produced *The selling weighed on the broader Tokyo Stock Price Index of all issues* from *The selling weighed on the broader Tokyo Stock Price Index of all issues listed on the first section.*

4.5 Getting More Named Entities

There is a certain tension between the drive towards Named Entities and Information Maintenance: a sure way to fail the latter is to perform a resolution of an anaphoric expression to a wrong Named Entity.

The current implementation is rather conservative, attempting resolution of just *he, his, him, her, she* to antecedents that are Named Entities. This task was shown to be within reach of a shallow resolution method with a success rate of almost 80% [3], as opposed to lower than 50% success rates for *it*.

We implement salience based anaphora resolution, maintaining two stacks of person names found in the text, one for each gender. The stacks are updated when a person is mentioned – by a full name, a partial name or a pronoun; Appendix A describes the algorithm.

Out of the 20 such pronouns in the TestSet sentences, 16 were resolved correctly. 3 mistakes were due to the missed reference with a common noun (ex. *his* in *The king wanted to convey his wishes ...* was resolved to a proper name from the previous sentence, rather than to *the king*). One mistake was due to our algorithm: *his* is resolved to *Andre Agassi* in *... Agassi said, congratulating Kroslak for his performance in the match ....*

The EAS-construction algorithm also substitutes partial names with full names; this occurred 5 times in the 123 TestSet EASes, all of which were correct.

4.6 Example

As an example of EAS construction, let us consider a sentence from the extract from Harriet Beecher Stowe's biography presented earlier (section 3). This name is the top one on the female names stack when we get to this sentence.

Example Sentence. Moving to Cincinnati with her father in 1832, she began to write short fiction.

Figure 1 presents MINIPAR's analysis of this sentence. Solid arcs represent dependency links; dashed ones - antecedence (*same-entity-as*) links. Labels on the solid arcs correspond to the dependency relations (for example, *subj, obj, aux, mod*). Every node consists of: the lexical string, including () for the empty string; base form (*move, begin*) and syntactic category information (*V(erb), N(oun), fin(ite) C(lause), A(djective)*); semantic information (tense).

From the sentence above, we automatically construct the following EASes:

1. Harriet Beecher Stowe moved to Cincinnati with her father in 1832.
2. Harriet Beecher Stowe began to write short fiction.
3. Harriet Beecher Stowe wrote short fiction.

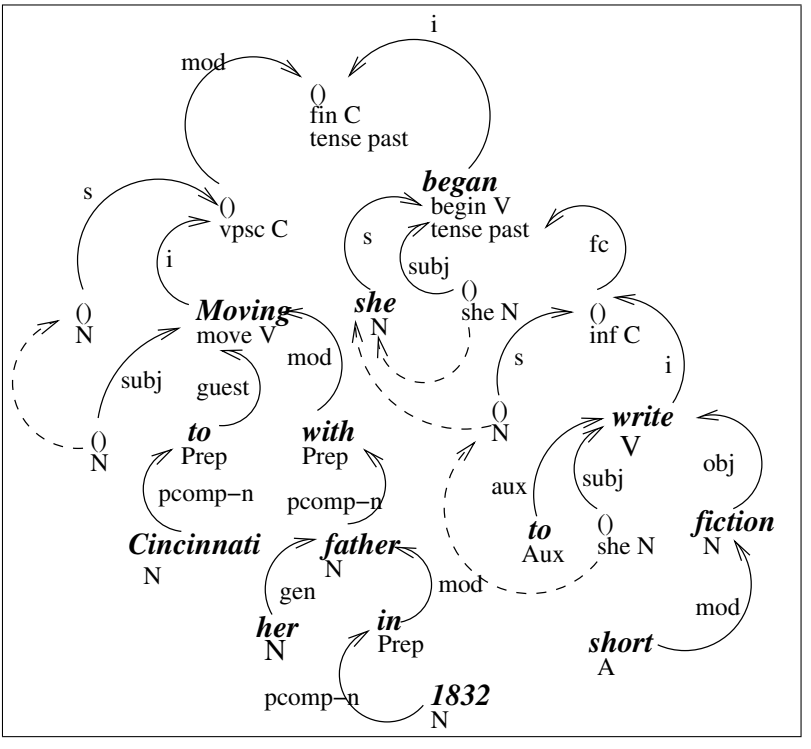

Fig. 1. MINIPAR's analysis of the source sentence

To generate (1), we determine that *moving* is not in a semantically problematic environment. It gets assigned the tense of its closest tensed governor, which is the past tense of the clause with the head *began* (the topmost node in Figure 1). Since the antecedence links for the subject of *moving* do not lead to any lexically realized string, we default to the subject of the clause of which the current clause is a modifier (see the left topmost dependency link labeled *mod* in Figure 1), which yields *she*. We then resolve the pronoun to the top of the stack. Getting to the pronoun *her*, we check the configuration and see that it is a possessor entity modifying a dependent of the main verb[8]. Since the subject of the verb is resolved to the same entity, we do not substitute the full name for *her*, as the corresponding Named Entity already appears in the clause.

In (2), *began* passes the semantic check. Since it is already tensed, no tense assignment is performed. Dependents are collected from the dependency structure, and the pronoun is resolved to the topmost element in the female names stack.

During the construction of (3), *write* is submitted to the semantic test. Since the governor *began* is not in the list of verbs that do not presuppose their complements, the test is successful[9]. Tense is again taken from *began*, and the an-

[8] Here dependence is mediated by the preposition *with*.

[9] The test would have failed had the sentence had *wanted* instead of *began*.

Table 1. Precision of EAS construction algorithm

Requirement	Met (%)
S-level Entity	112 (91%)
Single Verb	118 (96%)
Info-Maintenance	69 (56%)
1-3 together	68 (55%)

Table 2. Split of Information Maintenance Mistakes

Mistake	Made by (%)
Wrong verb	8 (6.5%)
Wrong tense of the right verb	10 (8.1%)
Comes from a bad sem. environment	5 (4%)
Wrong subject of the verb	16 (13%)
Wrong other dependent of the verb	26 (21.1%)
Wrong pronoun resolution	1 (0.8%)

tecedence links provided by the parser help us identify the subject, which is resolved to Harriet Beecher Stowe.

5 Testing the Algorithm

We use TestSet to evaluate the precision of the EAS construction algorithm. Out of the 123 sentences, 68 passed EAS requirements 1-3 (55%). Table 1 shows the detailed breakdown, with absolute numbers and percentages of EASes meeting the relevant criterion.

Table 2 shows the breakdown of Information Maintenance mistakes. For each mistake, the number and percentage of EASes that committed it are shown; if a certain EAS contained two different mistakes, it was counted twice.

We note that only one EAS actually had a wrong name substituted for a pronoun. The evaluation of the pronoun resolution algorithm reported in section 4.5 was performed running the system in the mode that just resolves pronouns. Hence, it tried to resolve all the relevant pronouns[10] in the texts, even if, when run in the EAS-construction mode, no EAS would have been produced from a certain sentence with a pronoun, or the pronoun would not have been substituted in an EAS (ex. *her* is not substituted in *Jane loves her mother* if the resolution is *Jane*).

To estimate the recall of our system, we asked 5 people to generate single verb sentence from an extract from Bertrand Russell's biography (the text and the exact wording of the instructions we gave to the examinees can be found in Appendices B and C, respectively). Our EAS-construction software and the 5 humans cumulatively produced 121 candidate EASes from the 7-sentence text. We then asked two other humans to judge whether each of these 121 can be inferred from the text.

[10] His, him, he, she, her

Next we identified 31 EASes that were produced by at least 3 humans; all of these were marked correct by both judges. We consider this set to be the gold standard set, since some of the EASes produced by just two humans were rejected as incorrect inference by one of the judges. Appendix D reproduces these 31 sentences.

Out of these, our EAS-construction software produced 10 (see Appendix D). It produced one additional EAS that was generated by two humans and marked correct. It also constructed 3 EASes that were not generated by any human but considered correct by both judges. Finally, 4 sentences were produced just by the software and marked as incorrect by both judges.

6 Discussion and Future Work

In this paper, we defined the notion of Easy Access Sentence - a unit of text from which the information it contains can be retrieved by relatively simple means, built to process single verb sentences with named entities. This is an attempt to mediate between the information-rich natural language data and applications that are designed to ensure the effective use of canonically structured and organized information, which is, however, hard to obtain without extensive human intervention.

We identified challenges in producing such middleware, the most difficult being the requirement to maintain the factual information encoded in the original text. This means both not to over-produce (avoiding non-factual constructions, like conditionals and domains of belief and desire verbs) and not to miss information (trying to consolidate into one fragment information that is dispersed in the original text, by resolving anaphora and retrieving covert subjects of verbs).

The small-scale evaluation of our implementation of EAS-production suggests that precision and recall figures are not yet satisfactory, estimated at 50% and 30%, respectively. While this might already turn out to be useful for some applications, our first objective is improving the performance of the algorithm. Error analysis showed that many mistakes are due to the dependency parser we employed (MINIPAR); using additional parsers and combining their analyses by a weighted vote might improve the reliability of the parse. In addition, whereas some disambiguation procedures we employed work well (anaphora resolution, name substitution), others need further analysis from the lexical semantic perspective - for example, the tense assignment procedure does not take into account the semantic behavior of the governor, and produces the correct result only in 65% of the cases. Finding conservative procedures for resolving definite noun phrases to named entities would also improve the EAS-hood of the system's output.

Acknowledgements. We would like to thank Lara Taylor and Jerry Hobbs for useful suggestions; Franz Josef Och, Dragos Stefan Munteanu, Eric Melz, Hal Daume, Mark Sprangen, Jonathan Graehl for their help in evaluating the system's performance.

References

1. Collin F. Baker, Charles J. Fillmore and John B. Lowe. 1998. The Berkeley FrameNet project. *Proceedings of COLING/ACL'98*. http://www.icsi.berkeley.edu/framenet/
2. Daniel M. Bikel, Richard Schwartz and Ralph M. Weischedel. 1999. An Algorithm that learns What's in a Name. *Machine Learning*, 34:211-231.
3. Kalina Bontcheva, Marin Dimitrov, Diana Maynard, Valentin Tablan and Hamish Cunningham. 2002. Shallow Methods for Named Entity Coreference Resolution. *Proceedings of TALN'02*.
4. John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin and John Tait. 1999. Simplifying Text for Langauge-Impaired Readers. *Proceedings of EACL'99*.
5. R. Chandrasekar, Christine Doran and B. Srivinas. 1996. Motivations and Methods for Text Simplification. *Proceedings of COLING'96*.
6. R. Chandrasekar and B. Srivinas. 1997. Automatic Induction of Rules for Text Simplification. *Knowledge-Based Systems*, 10:183-190.
7. Siobhan Devlin. 1999. Simplifying natural lanauge text for aphasic readers. *PhD dissertation*. University of Sunderland, UK.
8. Dekang Lin. 1998. Dependency-based Evaluation of MINIPAR. *Workshop on the Evaluation of Parsing Systems*.
9. Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin and Dragomir Radev. 2004. A Smorgasbord of Features for Statistical Machine Translation. *Proceedings of HLT/NAACL 2004*.
10. Joel Tetrault. 2001. A corpus-based evaluation of centering and pronoun resolution. *Computational Linguistics*, 27(4):507-520.
11. E. M. Voorhees and Lori P. Buckland (Eds) 2002. *Proceedings of the 11th Text REtrieval Conference*.

A Anaphora Resolution

Two gender stacks of Named Entities are maintained and reset for every text. We approximate the grammatical roles hierarchy (subject > object > indirect object > modifier) by the linear order of the constituents[11]. We proceed as follows:

- Upon hitting a name N
  - If N repeats[12] a name already in one of the stacks, extract it from the relevant stack unless the previous mention was in the same sentence. If so, do nothing more.
  - If N repeats a name, push N underneath names last mentioned in the current sentence that are also repeated names, but on top of new names in the current sentence and names last mentioned in the previous sentences.

[11] Tetrault's Left-to-Right Centering [10] performed very similarly with syntax-based and surface-based ordering - see comparison of LRCsurf and LRC therein.

[12] Just surname or just first name repeat a full name, unless there are different names with the same surname in both gender stacks - then the surname is rendered ambiguous and no substitution is performed.

  - If N is new, push N underneath names last mentioned in the current sentence, but on top of names last mentioned in the previous sentences.
  - If the gender of N is unknown[13], push to both stacks.
- Upon hitting a pronoun P
  - Resolve P to the *target name*, which is the top of the gender matching stack, unless P is accusative (him, her), and the subject of the verb is same-gender pronoun or a proper name; then *target name* is second in stack.
  - Update mention of *target name* with the current sentence number.
  - If *target name* is second in stack and the subject was a proper name, move *target name* to the top of the stack[14].
  - If *target name* appears in both stacks, extract if from the opposite gender stack.

B Bertrand Russell's Biography

Bertrand Russel, a philosopher and mathematician, was born in Trelleck, Monmounthshire, in 1872. He studied in Cambridge, where he became a fellow of Trinity College in 1895. Concerned to defend the objectivity of mathematics, he pointed out a contradiction in Frege's system, published his own Principles of Mathematics (1903), and collaborated with A N Whitehead in Principia Mathematica (1910-3). In 1907 he offered himself as a Liberal candidate, but was turned down for his "free-thinking". In 1916 his pacifism lost him his fellowship (restored in 1944), and in 1918 he served six months in prison. From the 1920s he lived by lecturing and journalism, and became increasingly controversial. One of the most important influences on 20th century analytic philosophy, he was awarded the Nobel Prize for Literature in 1950, and wrote an Autobiography (1967–69) remarkable for its openess and objectivity.

C Instructions to Human Generators

We have lately been working on software to make natural language texts simpler and more explicit. Our system currenly performs rewrites of the original sentences into sets of "factoids": subject-verb-object (possibly with some modifiers) assertions that the sentence makes.

We would like to ask for your help in evaluating the system. We would provide you with a text, and ask you to write down, for each sentence, the simple SVO factoids that you believe to be explicitly and implicitly asserted in the sentence. We are interested in generating factoids that depart as little as possible from the wording of the original texts. That is, we are interested in factoids that can be obtained from sentences via word/phrase deletions and some minimal rewriting. We are not after generating "Close the window" from "It is cold

[13] Lists of male and female first names are maintained; we thank Ulf Hermjakob for making these available to us.

[14] Pronominalization is a stronger salience marker than subject mention.

here". The rewrites below may help you internalize at the intuitive level the factoid definition we are after[15].

D Gold Standard Rewrites

Bertrand Russell was born in 1872 (5)[16]. Bertrand Russell was born in Trelleck, Monmountshire (4). Bertrand Russell was born in Trelleck (3). Bertrand Russell was born in Trelleck in 1872 (3). Bertrand Russell was born in Trelleck, Monmountshire, in 1872 (3). Bertrand Russell studied in Cambridge (5*). Bertrand Russell became a fellow of Trinity College in 1895 (5*). Bertrand Russell became a fellow of Trinity College (4). Bertrand Russell was concerned to defend the objectivity of mathematics (5). Bertrand Russell pointed out a contradition in Frege's system (5). Bertrand Russell published Principles of Mathematics (5). Bertrand Russell collaborated with A N Whitehead in Principia Mathematica in 1910-3 (5). Bertrand Russell published Principles of Mathematics in 1903 (4). Bertrand Russell collaborated with A N Whitehead (4). Bertrand Russell collaborated with A N Whitehead in Principia Mathematica (3*). Bertrand Russell offered himself as a Liberal candidate in 1907 (4*). Bertrand Russell offered himself as a Liberal candidate (4*). Bertrand Russell's fellowship was restored in 1944 (5). Bertrand Russell served six months in prison (4*). Bertrand Russell served six months in prison in 1918 (4*). Bertrand Russell was pacifist (3). Bertrand Russell lost his fellowship (3). Bertrand Russell lived by lecturing and journalism from the 1920s (5*). Bertrand Russell became increasingly controversial from the 1920s (5*). Bertrand Russell was one of the most important influences on 20th century analytic philosophy (5). Bertrand Russell was awarded the Nobel Prize for Literature in 1950 (5*). Bertrand Russell was awarded the Nobel Prize for Literature (5). Bertrand Russell wrote an Autobiography from 1967 to 1969 (5). Bertrand Russell's autobiography is remarkable for its openness and objectivity (5). Bertrand Russell was awarded the Nobel Prize (3). Bertrand Russell was awarded the Nobel Prize in 1950 (3). Bertrand Russell wrote an Autobiography (3). Bertrand Russell's autobiography is remarkable for its openness (3). Bertrand Russell's autobiography is remarkable for its objectivity (3).

[15] There followed an example with EASes generated by one of us from the biography of Harriet Beecher Stowe.

[16] The numbers in brackets show the number of humans who generated the sentence. An asterisk marks sentences generated by the software.