Tracking Information Extraction from Intelligence Documents

Christopher Welty[1]*, J. William Murdock*[1]*, Paulo Pinheiro da Silva*[2]*, Deborah McGuinness*[2]*,*
David Ferrucci[1]*, Richard Fikes*[2]*,*
[1]IBM Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532, USA
{murdockj,ferrucci,welty}@us.ibm.com
[2]Knowledge Systems Laboratory
Stanford University
Stanford, CA 94305, USA
{pp,dlm,fikes}@ksl.stanford.edu

Keywords: Novel Intelligence from Massive Data, Information Extraction and Link Analysis

Abstract

We describe here some of the research underlying the development of KANI (Knowledge Associates for Novel Intelligence), a hybrid system that combines large scale information extraction (IE) with knowledge representation (KR). The combination of these two technologies raises numerous research problems, such as an evaluation and understanding of the requirements that KR puts on IE and vice-versa, the identification of useful intermediate results in the process of transforming information from unstructured and massive collections into small and precise chunks suitable for automated reasoning, and how extend traditional explanation techniques to provide the ability to trace provenance of information through these transformations.

1. Introduction

A critical problem for supporting intelligence analysis over large collections of data is to provide that data in a form that is understandable, traceable, and capable of being used with advanced tools for searching, reasoning, viewing, etc. We describe here some of the research underlying the development of KANI (Knowledge Associates for Novel Intelligence), a hybrid system that combines large scale information extraction with automated reasoning and other technologies. In particular, we discuss the stages in a pipeline from unstructured data to actionable knowledge, and describe how the processing at each of these stages enables increasingly powerful analysis tools. We also describe how information flowing along this pipeline can be traced back to its sources.

2. Background

The KANI project began with the belief that one of the most difficult problems in knowledge representation (KR), the so-called "knowledge acquisition bottleneck," could be addressed by bringing to bear information extraction (IE) technology. At first glance, this appears a natural connection to make; IE systems focus on unstructured data (in our research we limit our examples to text, however significant progress in being made in analysis of other media) and produce some simple structure, whereas KR systems focus on structured data and produce inferred information or check consistency.

The combination of these two technologies, however, proved quite a challenge, and raised numerous research problems that have forced us to reevaluate each of the basic technologies. One very novel result, discussed in Section 3, is to consider the transformation of information from unstructured and massive collections into small and precise chunks suitable for automated reasoning, and to identify stages in the transformation that produce results in a form that can enable tools that increase in power and utility as information moves to successive stages.

Another novel result, discussed in Section 4, is an evaluation and understanding of the requirements that KR puts on IE and vice-versa.

Finally, as discussed in Section 5, we have had to significantly extend our explanation facilities to provide the ability to trace provenance of information through these successive stages.

3. The KANI Pipeline

Turning unstructured data into formally represented knowledge happens in successive stages in an information pipeline, each requiring more resources to produce, and enabling more powerful tools. We envision the analyst using the more advanced tools in each stage to further focus the processing for the next stage. In addition, at each stage the underlying formats, representations, and ontologies may be different, and we must be able to track the sources of information produced across these changes.

3.1 Keyword Indexing

The simplest and most scalable processing is the generation of an inverted index to support keyword search. Although techniques such as link analysis, query expansion, etc., can offer minor improvements, this approach is generally very low in precision. In addition to its current established usage, we consider the function of keyword search to be *domain corpus production.* We employ recall-improving techniques such as query expansion to reduce the size of the target corpus to the scale required by the next stage of processing (information extraction) – this is typically 1-2 orders of magnitude.

3.2 Information Extraction

Information extraction (IE) in general can be viewed as the analysis of unstructured information to assign labels (or *annotations*) that carry some semantics to regions of the data. The canonical example would be to label the text "George Bush" as Person. The field has advanced considerably since these beginnings, and are well represented by the ACE program (ACE 2004), participants in which produce annotations for entities (Person, Organization, etc.), relations (*partOf, citizenOf,* etc.), and coreference analysis.

While almost any kind of information processing can be folded into an information extraction view, in our system, IE components play the role of providing relatively shallow processing in order to be scalable. In particular, this stage limits itself to processing data in well-defined chunks (such as documents), and performs the same analysis on each chuck independently. As a result, IE processing scales linearly with the size of the domain corpus.

The value added by this stage of processing should be fairly clear: search queries can be augmented with more semantic information in order to increase precision. We have called this *semantic search*, and in addition to improved search capabilities, we consider the function of semantic search to be *focused corpus production*. The analyst uses the features of semantic search to further focus the corpus on relevant documents to the task at hand, in order to reduce the scale to that required by the next stage. This is typically another order of magnitude.

3.3 Coreference Across Documents

The annotations produced in the IE stage are used as input to *corpus-level processing*, the most important to our purposes of which is coreference analysis – the identification of individual entities that are mentioned (and annotated) in multiple places. Many of our IE components produce coreference analysis within chunks, but connecting these results across the entire corpus clearly requires processing that can collect information across the chunks, and thus will typically scale at a polynomial rate. In our experience, the most critical properties of co-reference are recognition of aliases and nicknames, common spelling variations of names (especially in other languages), common diminutives, abbreviations, etc.

Co-reference errors are probably the most noticeable of all the types of errors automated analysis produces. We have found that users expect the most common and well-known entities (e.g. countries, states, cities, and all their abbreviations, famous people and their nicknames, etc.) to have very high recall, and the least common entities to have very high precision.

The value added by this stage of processing is to enable queries on specific entities. While semantic search results are typically documents that must be read, co-reference analysis enables searching for extracted facts about individual entities (e.g. what relations have been extracted), and we call this *fact search.*

3.4 Knowledge Integration

Although it is not required, the data produced in the first three stages of our system are all based on the same underlying format (discussed in Ferrucci&Lally, 2004), which is simple extension of an OO programming model with a tight programmatic API and a loose semantics (that is, the semantics of a data model can be interpreted by software as the programmers choose). Using the information in more formal and general-purpose reasoning components requires a representation with a clearly specified semantics. In our work, we have focused on OWL, the first standardized KR language, as this representation.

OWL was a good choice for two primary reasons: it provides a natural basis for representing the kind of information being extracted from unstructured sources (entities and binary relations); as a popular standard, there are a wealth of tools available for it.

Moving the results of unstructured analysis into OWL is not a simple task, however. The semantics of the information extracted in the previous stages is not always precise, and the data itself is full of precision and recall errors. While we can hope to improve on the precision and recall of the previous stages, we take this problem to be fundamental: how can we make use of the results of IE in a formal reasoning setting? This question is addressed in detail in the next section.

The process of mapping the information from the previous stages in OWL is analogous to the general problem of semantic integration, and we have called it *knowledge integration.* The result of knowledge integration, an OWL knowledge-base that can be viewed as a graph, provides the ability to use OWL-based reasoning to perform more sophisticated *deductive search.* For example, we can express axioms of spatial or temporal contain-

ment in OWL, and conclude obvious (but nevertheless implicit) results, such as a person in Paris is also in France.
The knowledge integration problem ranges from logarithmic to (worst case) exponential in complexity, depending on the specific reasoning requirements.

3.5 Knowledge Selection

OWL has a number of expressive limitations, and in order to enable deeper and more sophisticated reasoning, we need to map our data from OWL into a more expressive language. In our work, we are using KIF, and thus another level of transformation is required. Clearly the additional expressiveness of KIF implies that the ontologies in KIF will be different than, or at least extensions to, the ontologies in OWL, potentially requiring further semantic integration as well as language transformation. While this is true in general, we have restricted our KIF ontologies to be axiomatic extensions to the OWL ontologies, and thus do not add any terms (classes or relations).
This requirement, that the OWL and KIF ontologies share vocabularies, introduces a special requirement for the handling of relations that, in our KIF ontologies, are higher arity than two, since OWL ontologies are limited to unary and binary relations. Our approach to this problem is described in (Welty & Fikes, 2005).
The reasoning a KIF-based representation enables is notoriously high in computational complexity and memory requirements. Indeed, there are numerous undecidable inference problems expressible in KIF, and as a result it is not possible to precisely determine the complexity of reasoning in general. In our experience, it is best to aggressively circumscribe problem spaces in order to ensure that any reasoning problems will be computable in a reasonable amount of time. To this end, we introduce a process of *knowledge selection*, in which the analyst interacts with the various search mechanisms to populate a *working knowledge base* of knowledge relevant to the current task.

3.6 Advanced Reasoning

The working knowledge base serves as a basis for viewing sections of the knowledge-base in OWL, and allowing the user to interactively refine and elaborate their knowledge of the current task, domain, or scenario. Advanced reasoning services requiring KIF are accessed by dynamically converting the selected OWL into KIF. Some of the advanced reasoning services we are developing are deductive query answering, temporal reasoning over the full Allen calculus, hypothesis testing, and context-based reasoning, among others. These are discussed elsewhere (Fikes, et al, 2005).

4. Requirements for Knowledge Integration

The central and novel problem introduced by moving information through these stages is knowledge integration: moving the results of unstructured analysis into OWL. Knowledge integration is difficult, and to our knowledge has not been done before, due to the vastly different requirements, and different communities, on each side. As a result, what seems on the surface to be a natural connection, that is producing structured representations from unstructured information and then reasoning over those structures, turns out to be a difficult challenge. While IE systems claim to produce “structure” where none existed before, the results, as produced, are not usable by traditional systems that require structure – such as databases and inference engines; the results require human interpretation. On the other hand, while KR systems claim to embody intelligent reasoning, they require as input the results of very sophisticated human processing by experts trained to see distinctions that would make Sherlock Holmes look simple-minded.
In order to reconcile these differences and bridge the gap, we are working in three areas: improving the previous stages to meet the basic requirements of reasoning, relaxing some the requirements of reasoning to make the KR-based components less fragile, and developing a component responsible for bridging the remaining gap using doses of automation and assisted human interaction. Below we list several critical differences and how we are addressing them.
Relationships. Simple IE systems that produce type annotations (such as Person, Organization, etc.) are not of much use as input to a reasoning system. These end up in a knowledge base as assertions that something is an instance of something else. There is very little reasoning that can be done with only that information. In order for reasoning to produce useful results, we need relationships to be extracted as well. For example, there is not much to conclude from the sentence, “Joe was in Paris,” if all that was produced was that “Joe” is a person and “Paris” is a place. In this case, a located-in relation would be useful as well, as simple spatial containment axioms plus basic world knowledge (e.g. that Paris is in France) would allow a reasoner to conclude that Joe was in France. We use a number of IE components that produce relations over text.
Annotations vs. Entities. In our experience, relation annotation by itself creates another problem. Every relation annotation creates a tuple whose elements do not appear in other relations, severely limiting the usefulness of reasoning, since the elements of the relation tuples are the *mentions* not the entities. For example, from the sentences, “Joe was in Paris. Fred was in Paris, too,” relation annotation would produce two tuples, however the elements of the tuples are not the strings, “Joe”, “Fred”, and “Paris”, but the regions containing those strings in the original text, and as a result we have four elements

identified by their position in text, *not* by their contents. Thus the first and second occurrences of "Paris" are different elements, and we could not conclude in a reasoner that, e.g. Joe and Fred are in the same place. In fact, without connecting these two mentions of Paris (both within and across documents), we end up with a large list of unconnected relation tuples. We address this problem with coreference analysis, making it a critical aspect of our system. In particular, consider that the output of knowledge integration is a graph – the graph without coreference analysis would be a disconnected set of connected pairs.

Precision. Formal reasoning systems are notoriously intolerant of errors, and IE systems are notoriously prone to producing them. In particular, logical reasoning becomes meaningless in the face of contradiction, most inference engines will prove any statement to be true if the knowledge-base is inconsistent to begin with. Although improving precision is an obvious approach to this problem, we take it as a given that IE processes will never be perfect, and furthermore even in the presence of perfect IE, data sources can contradict each other intentionally (e.g. reports from CNN and the pre-war Iraqi News Agency), and instead focus on making the reasoning systems more tolerant of errorful data.

Our simplest technique is to perform limited reasoning such as semantic constraints that can be checked rapidly, and that in our evaluations we find to be indicative of IE errors and not intended contradictions. This leads to a simple result in which processing semantics gives us an improvement in precision. For example, consider the sentence "Joe arrived at Bush Intercontinental." Several simple IE components have annotated "Bush Intercontinental" as a Person. A separate component annotates the *arrivedAt* relationship in this sentence between "Joe" and "Bush Intercontinental". Specifying and then checking a simple range constraint on the *arrivedAt* relation (that e.g. you cannot arrive at a person) would give us evidence to consider the person annotation incorrect.

Another simple approach is to use reasoning without contradiction, i.e. so-called "forward chaining". This allows us to infer the consequences of material implication axioms without checking for consistency. This can be handled in OWL by partitioning an ontology into two parts: one containing disjointness axioms, and one that is disjoint-free. In OWL, disjointness axioms are the only way to express a contradiction. For example, in the disjoint-free ontology module we could express spatial containment axioms (e.g. the *locatedIn* relation is transitive), allowing us to infer the transitive closure of spatial relations. Interestingly, this technique *decreases* overall precision as it tends to propagate errors, however it provides a useful increase in recall.

It is still the case that the more advanced reasoning services require absolute consistency in the knowledge-base. Another approach we take is to automatically *partition* the knowledge-base into smaller consistent chunks. There are a myriad of possible ways in which we can create these partitions. One is to add knowledge extracted from a single document at a time; as soon as a document is added that causes the partition to be inconsistent, remove the knowledge from that document and close the partition. This approach clearly requires special treatment for internally inconsistent documents.

Recall. Although recall problems in the extracted data do not impact the reasoning components, they do immediately impact the user (this is enhanced by the fact that, as discussed in the next section, all information is linked to its sources). We are developing an understanding for how to explore the tradeoff between precision and recall. We are considering that different tasks may require different tradeoff choices (i.e. to prefer recall over precision or vice versa).

It is also important to note that using inference can help improve recall, however it is a different sense than is typically used in IE measurements. Recall measurements are based on comparison to a "ground truth" (i.e. a human annotated corpus), in which implicit information does not appear. For example, in the sentence "Joe arrived in Paris", we would not expect a test corpus to include the relationship that Joe arrived in France, yet this inferred information clearly increases the recall.

Scalability. IE techniques scale far better than KR techniques, and as a result we also need to limit the amount of data that any reasoning component has to deal with. In our experience, documents provide an excellent and reliable heuristic for KB size, as well as for consistency. We have found that, excluding IE errors, about 90% of the documents we process are internally consistent, and thus far all documents (we focus mainly on news articles, intelligence reports and abstracts) have been the basis of small enough KBs for any of our advanced reasoning systems.

Still, document-based partitioning is inadequate for a lot of intelligence tasks. So we use a variety of incremental capabilities, such as adding all the knowledge extracted from the data sources concerning a single entity. Our envisioned use-case is that an analyst identifies a relevant document, creates a working knowledge-base from its extracted contents, identifies a relevant entity (e.g. a person or an organization of interest), and augments the knowledge-base with all the information about that entity that has been extracted from the corpus.

5. Information Pedigree

One of the primary requirements of our project is that all information be traceable back to its source. This requirement combined with our pipeline from data through analysis to reasoning forced us to reconsider some basic assumptions about how we explain, and present, provenance information to the analyst. Our existing explanation technology, Inference Web [], was originally designed to store logical proofs, whereas the pipeline added the demand for a complete trace of the processes that contributed to a piece of information's pedigree.

Recording such a trace of all the processes upon which information depends may be useful for a variety of reasons, e.g., explaining how results were obtained, vetting results from unknown, untrusted, or unreliable processes, and selecting follow-on tasks to perform. To be complete, such explanations and analyses should be based on integrated representations of all the stages in the pipeline described above. However, while representations of processes for reasoning over structured knowledge are fairly common (e.g., Davis, Buchanan, & Shortliffe, 1977; Swartout, Paris, & Moore, 1991; McGuinness, 1996), there is much less work on representing processes that extract structured knowledge from unstructured information.

One requirement for representing analysis of unstructured information is a *taxonomy* of analysis tasks that is complete enough to accurately describe task functionalities and abstract enough to hide unnecessary technical details. Given such a taxonomy, it is possible to observe the tasks that analysis systems are performing and record those observations. We have built such a taxonomy for the analysis of natural-language text. We have also implemented a system that records processes (using the terms of this taxonomy) in a manner that is consistent and compatible with explanations for processes that use extracted knowledge to perform additional reasoning. Our system uses these records to provide browsable explanations, and we intend to also explore other applications such as vetting results and selecting new tasks to perform.

We divide the process of extracting knowledge from text into three areas: *annotation*, *coreference resolution*, and *knowledge integration*. We have identified three distinct primitive tasks in each of these areas.

5.1 Annotation Inferences

The annotation tasks involve making assertions about individual spans of text (a span is defined by a beginning character position and an ending character position within a given document, e.g., characters 0-8 of *foo.txt*). The example in Figure 1 includes only annotation tasks. Below is our list of annotation tasks:

1) **Entity Recognition**: Concludes that a *span* (i.e., a segment of the original text) refers to an *unspecified* entity of a specified type; i.e. produces an *entity annotation.*

Example: Inferring that the span 0-16 (i.e., "Joseph Gradgrind") of Document 1 refers to an entity of type Person.

2) **Relation Recognition**: Concludes that a span refers to an unspecified relationship of a specified type; i.e., produces a *relation annotation*.

Example: Determining that the span 0-50 (i.e., "Joseph Gradgrind is the owner of Gradgrind Foods.") of Document 1 refers to a relationship of type ownerOf.

3) **Relation Annotation Argument Identification**: Concludes that an annotation fills a *role* of (i.e is an argument to) a relationship.

Example: Determining that the *subject* role in the aforementioned ownerOf relation annotation is filled by the aforementioned Person entity annotation.

5.2 Coreference Inferences

The coreference resolution inferences build on the results of annotation.

4) **Entity Identification**: Concludes that a set of entity annotations refer to a particular *entity instance*.

Example: Determining that the entity annotations on "Joseph Gradgrind" and "Joe Gradgrind" refer to the same entity instance.

5) **Relation Identification**: Concludes that a set of annotations refer to a particular *relation instance* of a specified relation type with specified values for its roles.

Example: Determining that the relation annotation on "Joseph Gradgrind is the owner of Gradgrind Foods" refers to a relation instance (OwnerOf UID1 UID2).

6) **Extracted Entity Classification**: Concludes that a particular entity has a particular entity *type*. Typically, the types assigned to the entity annotations (by Entity Recognition) are used to select a type for the entity; for example, a voting scheme may be used to select the most common type among the annotations for the entity.

Example: Determining that the type of the entity referred to by "Joe Gradgrind" is Person.

5.3 Knowledge Integration Inferences

The result of the knowledge integration process is a knowledge-base (KB) containing instances and relationships that were derived from the previous stages. There are three types of mapping inferences whose provenance is recorded:

7) **Entity Mapping**: Concludes that an entity instance in the KB is derived from a set of entities and relation instances. Note that it is possible for relations to map to entities or vice versa, depending on representation choices made. Note also that we do not record precisely *how* the source information was transformed into the KB, just what information was used to create the instance.
8) **Relation Mapping**: Concludes that a relationship in the target KB is derived from a set of entity and relation instances..
9) **Target Entity Classification**: Concludes that an entity instance is an instance of an entity type in the target ontology (based on the types assigned to the instances from which it was derived).

6.0 Conclusion

We have presented three of the main challenges we have encountered in attempting to build a hybrid system that combines information extraction (IE) and knowledge representation (KR) technologies, to address the so-called "knowledge acquisition bottleneck." Despite seeming like a natural combination, this proved quite a challenge, and raised numerous research problems that have forced us to reevaluate each of the basic technologies.

One very novel result is to consider the transformation of information from unstructured and massive collections into small and precise chunks suitable for automated reasoning, and to identify stages in that transformation that produce results in a form that can enable tools that increase in power and utility as information moves to successive stages. Another novel result is an evaluation and understanding of the requirements that KR puts on IE and vice-versa. Finally we have had to significantly extend our explanation facilities to provide the ability to trace provenance of information through these successive stages.

References

Automated Content Extraction (ACE) Homepage. http://www.ldc.upenn.edu/Projects/ACE/. 2004.

Davis, R, Buchanan, B., & Shortliffe, E. 1977. Production Rules as a Representation for a Knowledge-Based Consultation Program. *Artificial Intelligence*, 8:15-45.

Ferrucci, D. & Lally, A. 2004. UIMA by Example. *IBM Systems Journal* 43, No. 3, 455-475 (2004).

Fikes, R., D. Ferrucci and D. Thurman. The KANI system. Also Submitted to IA-2005.

McGuinness, D.L.. Explaining Reasoning in Description Logics. Ph.D. Thesis, Rutgers University, 1996. Technical Report LCSR-TR-277.

McGuinness, D.L. & Pinheiro da Silva, P. 2004. Explaining Answers from the Semantic Web: The Inference Web Approach. *Journal of Web Semantics* 1(4):397-413.

Mahesh, K., Peterson, J., Goel, A., & Eiselt, K. 1994. KA: Integrating Natural Language Understanding with Design Problem Solving. In *Working Notes from the AAAI Spring Symposium on Active NLP*.

Moldovan, D. & Rus, V. 2001. Explaining Answers with Extended WordNet. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, France.

Pinheiro da Silva, P., McGuinness, D. L., & Fikes, R. 2004. A Proof Markup Language for Semantic Web Services. *Information Systems Journal* (to appear)

Pinheiro da Silva, P. McGuinness, D. L., & McCool, R. 2003. Knowledge Provenance Infrastructure. *IEEE Data Engineering Bulletin*, 26(4), pages 26-32.

Ram, A. 1994. AQUA: Questions That Drive the Explanation Process. In *Inside Case-Based Explanation*, Schank, Kass, & Riesbeck (eds.), pp 207-261.

Swartout, W.R., Paris, C., & Moore, J. D. 1991. Explanations in Knowledge Systems: Design for for Explainable Expert Systems. *IEEE Expert Systems*, 6:3, 58-64.

Welty, C, and Richard Fikes. 2005. An OWL vocabulary for relationship reification. W3C working draft.