Indra: Emergent Ontologies from Text for Feeding Data to Simulations

Deborah Duong
Augustine Consulting, Inc.
Naval Postgraduate School
Monterey, CA
dvduong@nps.edu

Nicholas Stone
Virginia Tech
National Capital Region
Arlington, VA
nick.stone@vt.edu

Ben Goertzel
Novamente LLC
Rockville, MD
ben@goertzel.org

Jim Venuto
Jenzabar
Charlottesville, VA
zeus@vt.edu

Keywords:

Natural Language Processing, Social Role Networks, Ontology, Mutual Information, Information Extraction, Clustering

ABSTRACT: One of the major differences between irregular warfare (IW) simulation and conventional warfare simulation is the differing data requirements. The data required by conventional warfare simulation is very narrow and well defined compared to that required by IW simulation. In fact, almost any data that is remotely related to social conditions on the Internet and beyond may be used in IW simulation. There is no problem with the existence of data for simulations in this information age. The problem is that this data is not expressed in a way that can be used for simulations. This paper is about a natural language processing program, Indra, which works towards solving this problem by making data available for multiple purposes other than those for which it was first intended.

1. Introduction

Indra, a text extraction tool that creates ontologies of roles and role relations from unstructured text, was created for and used at US Army organizations INSCOM (Intelligence Command), I2WD (Intelligence and Information Warfare Directorate), RDEC (Research, Development and Engineering Center), and TRAC (Training and Doctrine Command Analysis Center). This paper describes the "unsupervised scatter-gather" clustering algorithm at the basis of Indra that facilitates the emergence of categorizations called ontologies from unstructured text, such as from a newspaper. This is a significant capability because it can take data expressed in one form, for the purpose it was made, and automatically express it another form for another purpose. It uses natural "co-locations," or correlations between the existence of data that falls into one category and data that falls into another, to create these groupings and to coerce them into a set of groupings other than it was created for.

This algorithm uses feedback on parsed Actor, Action, Object (AAO) triplets in text (what many think of as subjects, verbs and objects), to categorize the Actions based on the Actors and Objects they connect, and the Actors and Objects based on the Actions that connect them The ontology is constructed with information from "higher-order word co-occurrence." That is, an entity is categorized by not only the actions it takes, but the other entities that take those actions and the other actions that those entities take, and so on. All of this information is used to estimate the minimal entropy answer, the minimal description of the set of categories, the one that naturally makes the most sense given the data. The answers are in probabilistic form as well: instead of belonging to a category or not, a word or category has a distance from a "center" of a category. The categories represent concepts that exist in a "semantic space" where concepts have a distance from each other. This is similar to another natural language technique, Latent Semantic Analysis (LSA), which also creates a semantic space based on

higher-order word co-occurrence. Both find "bases," reducing the dimensionality of the semantic space, to order thought, and both are like the brain in their usage of higher order co-occurrence[1]. However, Indra uses words in parsed sentences, sentences in which the relations between Actors, Actions and Objects have been determined, to make its semantic space incrementally; while LSA forms this space all at once. Additionally, LSA does not use the information in the parse, because it is a "bag of words" technique. It is the incremental nature of Indra that gives it the additional property of being very flexible and able to conform to, as well as fill in the gaps of, existing ontologies. Similar to the way a child learns his parent's concepts despite relatively few explanations, Indra is designed to drive interpretation of data to a particular ontology with very few explicit points of correspondence given between ontologies. This property makes Indra a particularly good way of initializing and maintaining the data of social role network agent based simulations.

2. House of Mirrors Design Pattern

Indra is a system designed to create an ontology from unstructured text. There are systems that do something similar, such as Cimiano's[2], but Indra is designed to be flexible, and this flexibility is very important. Indra does not need many explicit extraction rules to match ontologies. To match a simulation ontology to the freetext data, the analyst can save the work of telling "why" an object belongs to a category, because only example groupings of entities and relations need be specified. Indra finds a data-driven ontology, by the use of statistical methods, that can be made to adapt to an existing hypothesis-driven ontology, with an arbitrary amount of input from that ontology. It can be made to fill in the gaps. Indra is of the "house of mirrors" design pattern, and is conducive to combined hypothesis-driven data-driven approach. That is, Indra may be used to take a data-driven ontology and combine it with a hypothesis-driven ontology, such as the ontology of a social theory or simulation. In fact, Indra is the name of the Indian god who had a "house of mirrors," a net with an infinite number of jewels that each mirrored all of the other infinite number of jewels.

In the "house of mirrors" design pattern, every element is defined by the other elements. In Indra, the most primitive levels of entities and actions between entities are put into a subsumption hierarchy of roles and role relations. When actors are defined by the actions they perform, they group into roles. Likewise, when actions are defined by the roles that perform them, they become role-relations. The roles are defined by the commonalities in the actions that entities engage in, and the role-relations are defined by the commonalities in the entities that the actions connect. For example, Indra might group boxers together because they engage in the same set of actions, such as "punching" or "trash talking." Indra would also group the actions that go with boxing together, into a "boxing" role relation, because these actions occurred between the same pairs of entities. This is circular logic, but this circularity is just what makes the system flexible.

To see why this is so, let us look at another "house of mirrors": the coevolving species in an ecosystem. If we look at modern ecosystems, we see many animals that originated in other ecosystems. However, introducing a species to a system is a difficult thing: it tends to either die out or take over, and usually is introduced with several other species from its former ecosystem at the same time, at which time it usually forms a microcosm. However, mixing species is not a hopeless cause, as evidenced by the fact that the species we see in modern ecosystems almost all came from elsewhere, and were successfully merged in. The key is that they have arrived at the right time; at the point of their introduction, they were species whose time had come. At that point, the ecosystem that they were introduced to adjusted to them. If they had particular traits such as fruits, then birds would evolve to depend on these fruits, and other species in the ecosystem would receive selective pressure to have such fruits. In other words, if something exists in a house of mirrors, then everything else in the house of mirrors will adapt to it, and vice versa, until it becomes a necessary part of the system.

In terms of Indra, the same principle applies to ontologies. If we give Indra groupings of words, say nouns, that are together in an ontology, Indra will find the collocated words, say verbs, that reinforce the existence of that grouping. These collocated words will be consonant with the groupings, will reinforce them, and will serve to find more words that belong in those groupings. Further, they will serve to find more groupings, than those collocated words connect. For example, given texts about families and names of fathers grouped together, Indra will find relations to other family members, including sons. Once it has found the relations to sons, it can group sons together. And once it found those, it can group together relations to siblings, etc, all of which are consonant with the concept of fathers. Having the relations that fathers are engaged in helps Indra to find more fathers as well. If Indra is given a more detailed set of groupings, it will fill in the gaps.

What happens when we give Indra groupings that contradict the data? The same thing that would happen to a species that you introduced to an ecosystem at the wrong time: either the species and the ecosystem adapt to each other, or the species would die out. It depends on just how contradictory they are, and if a middle ground may be found. The groupings that the analyst gives will be changed to match the data; for example, when collocated

actions of father groupings are found, and the father groupings are reinforced by them, then the members of the grouping that are not fathers will be removed, and other fathers added. Conversely, if the analyst's grouping is correct but the data is arranged incorrectly, the analyst's grouping can improve the data's arrangement. For example, suppose that the reason that collocations for "father" cannot be found is that the parser parsed them incorrectly. The analyst's groupings can help Indra to find the correct parse, so that the correct collocated actions are found for fathers. What is correct wins in the end, because support for correct grouping actually exists in the set of alternative parses, and if the grouping was incorrect, then there would be no alternative parse to support it. Because Indra is a house of mirrors, correct groupings help to find correct parses and vice versa.

3. Iterative Feedback

There are two kinds of feedback that Indra is designed for: "side to side" feedback, which is present in Indra now, and an "upper-lower" feedback, which is soon to be implemented. Side to side feedback was illustrated in the above example of how knowledge of the grouping of fathers can lead to knowledge of the grouping of siblings. The assignment of individual entities in documents to groups based on one of its links affects the assignment of the other links, and the assignment of those links affects the assignment of the other entities, and the whole interconnected group comes to a consensus. This is because the basis of assigning an entity to a group is the set of actions that connect to it, and the basis of assigning an action to a group is the set of entities that connect it.

Upper-lower feedback was illustrated in the example of how the correct groupings can serve to correct the parse. We can say that entities and actions are at the same level of meaning, and their effect on each other results in a global consensus on that level. The groupings of entities and links is the "word sense" level of meaning, and is the middle level of meaning in the Indra system. The lower level of meaning is that of the parse. A parsed path is used for the actions, and the parse affects those actions. At the same time, the parse is chosen based on the groupings of the entities and actions. Parses are chosen such that they reinforce the groupings that already exist; for example, knowledge that a salad is food , croutons are a food, and a fork is a utensil enable us to parse "John ate the salad with croutons" such that "with" modifies "salad" and "John ate the salad with a fork" such that "with" modifies "ate".

The upper level of meaning in Indra is the arrangement of the word sense groupings into a subsumption hierarchy. Indra will actually form two subsumption hierarchies, one for roles and one for role relations. These will be created by grouping role groupings according to their role relation context, and grouping role relations groupings according to their role contexts. These groups of groups form a subsumption hierarchy, which feeds back to word assignments on the middle level of meaning. For example, an upper level concept of a mammal is needed to pull "dolphin" out of the same group as "tunafish" and into the same group as "cow". Just as the parse, the lowest level of meaning, is chosen to reinforce patterns found at the middle level of meaning, the word sense groupings, so is the middle level meaning readjusted to reinforce the patterns found at the upper level of meaning, the ontological level.

The feedback (side-to-side and later, upper-lower) gives Indra at least three advantages over other programs that find ontologies in data. First, it is designed to be more accurate. It settles down on a global consensus on meaning, taking context into account to a greater degree than other natural language processing programs. Second, it is designed to be flexible, so that its ontologies can be aligned with very few points of correspondence to other ontologies. Finally, Indra's use of a generic measure of similarity based on information theory and the way the mind works, mutual information (MI), facilitates the inclusion of other modalities such as images and video, so that linguistic and nonlinguistic data may be fused[3] .

4. Indra's Design

Indra is a clustering algorithm, but it is different from traditional hierarchical clustering algorithms, in a way so as to form an ontology. The major difference between the Indra algorithm and other clustering algorithms is that there are two ontologies, each of which forms the context for the other, so that the hierarchy is formed by taking them in alternation. The entities group the links and the links group the entities. Therefore the hierarchical clusters are nonlinear, not all there at first, but formed together. In terms of ontologies, one hierarchy of classes is formed, and another hierarchy of properties is formed.

It is possible to do this with an algorithm that doesn't deal with word sense. However, it is also possible to take an algorithm that does, that re-forms clusters based on the larger picture of how they behave together. When we see the bigger picture, with the context of the higher level of description, we are better able to see that the part of one cluster really went with another part, and so we regroup it. When the larger picture changes very much, it is even more important to go back and reform clusters. The unsupervised scatter-gather algorithm enables Indra to revise and reform clusters based on new knowledge gained from structures at other levels.

4.1. Overall Structure

First a corpus, such as Reuters, is sent through a series of methods that find the mutual information values between

the words in AAO triplets (actor action object), or words and documents, depending on the program mode. "Mutual information" is a general concept from information theory which tells how uniquely associated one sign (or word in our case) is with another, and in natural language processing is used to group words together that have similar relations to other words[4]. Then, the ontology builder operates on the mutual information data to build up an ontology. Finally, the ontology is created, with dictionary words closest to the centroids used as labels. Instead of crisp properties, the probabilistic ontology has scalars associated with each property to fill in with either the mutual information scores, or probabilities. In future iterations, this ontology will be read into a probabilistic Web Ontology Language (OWL) file [5].

4.2. Frequency Counter Method

The purpose of the frequency counter method is to count and create text files that describe the co-occurances between the entities and the actions, from which the mutual information tables will be computed. All the documents are sent through the frequency counter. Only sentences marked by the AAO parser with a verb, an entity in the subject, and an entity in the object are used. When such a sentence is found, its verb and entities are normalized. The verb is stemmed The name of coreferenced entities, both orthographic and pronominal, are to be changed to the first name and last name that they represent. This name will be decided on by the most common first name and last name of the coreference chain. Entity type information is retained, so that a person with the same name as a location remains separate.

Once the verb and entities are normalized, the frequency counter counts the number of occurrences of an AAO triplet per document, and these are saved.

4.3. Mutual Information Calculator Method

The purpose of this calculator is to compute the mutual information between entities for the ontology builder to use.

For now, the mutual information calculator outputs three flat files, containing the mutual information values between actors and actions, the mutual information between actions and objects, and the mutual information between actions and actor-object pairs. The mutual information is calculated as in Pantel and Lin[6], as follows:

$$mi_{w,c} = \frac{\frac{F_c(w)}{N}}{\frac{\sum_i F_i(w)}{N} \times \frac{\sum_j F_c(j)}{N}} \qquad (1)$$

Here, w is "word", c is "context" and N is the total of all entries in the frequency count table. $F_c(W)$ is the frequency of a word in a given context. For us, the word is the action , and the context is the entity as an actor, as an object, or both, depending on which mutual information output file is being created. For the actor-object file.

$SUM_t(F_{ct}(W))$ is the frequency of a action in any context. For all three mutual information files, that would be all the entities that co-occurred with an action, both actor and object. $SUM_j\ F_c(W_j)$ is the frequency of all of a single context, and differs for each file. For the actor-object file, it is the sum of the row for the action for a given actor-object pair. For the actor table, it is the sum of the row of the action for a given entity as an actor, and for the object table, it is the sum of the row of the action for a given entity as an object.

Then, after MI is calculated, it may be multiplied by the following discounting factor, so that it will not magnify noise [6].

$$\frac{F_c(w)}{F_c(w)+1} \times \frac{min\left(\sum_i F_i(w), \sum_j F_c(j)\right)}{min\left(\sum_i F_i(w), \sum_j F_c(j)\right)+1} \qquad (2)$$

4.4. The Ontology Builder

The ontology builder implements an unsupervised scatter-gather algorithm. There are two hierarchies output, one for roles, that comes from the classifications of actors and objects (entities), and one for role relations, that comes from the classifications of actions(aka verbs, parse paths, or links). If the roles are put into an ontology as the classes, then the role relations are the properties that other roles fill. Each contains all of the mutual information based groupings, and is used to classify the social network, but is not itself the social network. For example, the role (class) hierarchy does not contain a node which represents an individual, but it can be used to compute which references in text refer to particular individuals. For example, the ontology may contain the class "Boxer"

and the actual data in the files from which a social network is extracted may contain the individual (entity extracted from text) "Mohammad Ali." The ontology may classify " Mohammad Ali" as a "Boxer" because in the texts, he does what boxers do, as indicated in the MI scores associated with the properties. The MI scores for the properties may be converted to Bayesian conditional probabilities, , in which case a probabilistic ontology may be generated.

Output ontologies approximate the minimum entropy/maximum mutual information grouping of the input, to make the most likely language model. Their nodes form a set of trees. The role is labeled by the n instances, that occur in text, closest to the node's centroid. The ontology is a tree rather than a directed acyclic graph (DAG) because, if the role should really have more than one parent, it divides into "senses" of the role. For example, if the role "mother" is part disciplinarian, and part teacher, and part caretaker, then we might have nodes mother1, mother2, and mother3 that are child nodes of the nodes disciplinarian, teacher, and caretaker respectively (of course, the nodes disciplinarian, teacher and caretaker could also end up being divided). Any DAG may be turned into a tree in this manner, and a small change to the algorithm to turn off node splitting will make it a DAG. How wide the distinctions are drawn is determined by the parameters of the clustering algorithm. However, the grouping will approximate the most probable, maximum mutual information grouping of the input given the clustering parameters.

At the lowest level of the role ontology is the new grouping of the word sense, which corresponds to an individual actor or object, that represents an person, or in natural language processing terminology, an entity. For example, "Mohammad the boxer" is one "sense" of "Mohammad" and "Mohammad the accountant" is another. On the levels above the entities are roles, in a subsumption hierarchy. A "role" is just a grouping that includes a list of entities or roles that belong to that role. If an individual in reality belongs to more than one role, then the entity which represents it in the document is be assigned to a particular role depending on its context in the document. For example, "Arnold Schwarzenegger" would be put into an "acting" role in entertainment contexts, and in a "governor" role in political contexts. That is the flip side of having the ability to differentiate persons based on roles: If Indra is used to classify names into persons, one person may be classified incorrectly as two different persons just because he has two different roles that are not often seen together.

The role relation tree now just classifies verbs for actions, but will contain full parse paths in a future version. In the prototype, this path is a single word, an AAO "action" verb. In subsequent versions, the path will be a complete parse path between a subject entity and an object entity. It could be a learned path, or a path from a parser that gives us alternative paths to choose from (so that it accepts feedback). A path is the sequence of words that connect two entities that exist in the same sentence. A linguistic, or parsed path, is used, whether a simple verb as in the prototype, or a complex path as in subsequent versions that include feedback to the parser. For example, in the prototype, the sentence "John always speaks to Jill" the path is "speaks" and that path connects "John" and "Jill." In the next version, that includes paths, the path would be "speaks to" rather than "always speaks to" The lowest level of the role relation hierarchy would be the concept denoted by a path or verb. Above the lowest level, the role relation hierarchy will contain role relations. A "role relation" is a grouping that includes a list of the paths or role relations that belong to it. This grouping serves to normalize paths, which is important to browsing, keyword searching, and asking questions of documents.

Original actors and objects are kept track of according to the document they come from, so that when they are reassigned to other clusters, every instance in the same document is reassigned.

A doc instance is for actor and object types only. It points to all of the triplets of a doc that have the Entity as the object and all of the triplets of a doc that have the entity as the actor. A triplet can appear in one or two doc instances. When an entity is subtracted out of another, entire doc-instances leave.

4.4.1. Objects that the Ontologies are Composed Of

Triplet.

Document//String, original document id that the triplet occurred in

Action//String, from document, normalized

Actor//String, from document, normalized

Object//String, from document, normalized

numOccurances//int

ActionNode//Node, the Classification to which this triplet maps for Action

ActorNode//Node, the Classification to which this triplet maps for Actor

ObjectNode//Node, the Classification to which this triplet maps for Object

DocInstance

Name//String, An Entity, Normalized representation of an actor or object that appears in a document

Document//String, original document id that the triplets occurred in

Position// Actor Object

Triplets//Triplets in the document that contain the entity

Node

Triplets //HashSet<Triplet>, the triplets that this Node encompasses if it is a leaf

Children//HashSet<Node>, the subsumed Nodes (empty if a leaf)

Parent//Node, the parent Node (empty if a root)

Role extends Node

Actors//HashMap<RoleRelation, double>, MI values of Nodes from the Actor file

Objects//HashMap< RoleRelation, double>, MI values of Nodes from the Object file

RoleRelation extends Node

ActorsObjects// HashMap< Role, double>, MI values of Nodes from Actor-Object file

Ontology

Roots//HashSet<Node>, the root Nodes

4.4.2. Ontology Builder Main Loop: the Unsupervised Scatter-Gather Algorithm

Each role node (or, in terms of an ontology, class) contains two feature vectors that represent contexts on the role relation (property) hierarchy, and each role relation node contains a feature vector that represents contexts on the role hierarchy. The two feature vectors in the role nodes represent the role relation (actions) the role actively takes, and the role relation (actions) that a role passively accepts. The feature vector in the role relations nodes represent the pair of roles (actor and object) that have it as the role-relation. These contexts are used to compute similarity: for the role hierarchy, similarity with other roles, and for the role relation hierarchy, similarity with other role relations. These similarities are used to group the nodes in their parent nodes.

The main loop of the ontology builder is

1. Alternate Ontology. There are two ontologies to be made, one for roles (entities) and the other for role relations (links). Initialize or take the opposite of the one chosen last iteration. Each ontology is composed of classes which are the properties (feature vector) of the other ontology. So, if the Roles are chosen, then they are the Concepts, and the Role Relations are the Properties; and if the Role Relations are chosen, they are the Concepts, and the Roles are the properties.
2. Compute Mutual Information. Compute the MI score of every Concept to every single other Property, to be the feature vector of the Concept in a space defined by the Properties. This is done with the most currently done classifier of the Property, as it exists in the three current elements of a triplet. On the first iteration, use the normalized words in the text as the Property. On subsequent iterations, the property is the latest grouping of the word into the clusters of the opposing ontology, that each individual triplet was classified into in step number 8.
3. Cluster. Use a standard clustering algorithm to cluster the vectors. Those that may be set to differentiate small tight clusters in which many fall within a smaller radius are preferred. Some concepts will categorize as within a new larger concept, and the rest as outside the concept. Those outside of the concept are outliers.
4. Split. For every Concept outside the new larger concept (outlier), take every doc-instance assigned to it and find its distance from an established cluster. If over n percent of the doc instance triplets going to a single other Concept, split the outlier by taking the doc-instance triplets from the outlier and entering them into the space as a separate concept. The other half of the split is made from the remaining triplets.
5. Re-cluster. Now that new concepts exist, re-perform clustering. The newly split concepts may become part of new clusters, and some parts of clusters may also become new outliers.
6. Assign outliers to their own clusters.
7. Name the clusters with the original text words of the triplets nearest to the centroid.
8. Assign the words corresponding to the current ontology in the original triplets to the concepts of the new clusters, making note in the triplets and in the clusters. There should be a list of concepts for each position in the triplet.
9. If there are no more outliers remaining, or the only changes are repetitive, stop. Else go to step 1.

Upon completion of this loop, we should have an approximation of the maximum mutual information grouping. It is finding bases as is LSA, but using a mechanism that is closer to the way the mind does the same thing, and is the only method that truly takes syntax into account in doing so.

4.4.3. Similarity Calculation in the Ontology Builder

The actor file and object file are used to calculate the role ontology. The actor-object file is used to calculate the role relation ontology. Every normalized noun may be characterized by a context of normalized verb/verb path relations. We can say that the normalized noun has a feature vector of normalized verb/verb paths which it is the subject of, and a feature vector of normalized verb/verb paths which it is the object of. These normalized verb/verb path contexts are slots and the mutual information (from the actor and object files) are the values of the slots. The similarity between two entities may be computed by the cosine coefficient of their respective mutual information vectors (equally weighted) [6]:

$$sim(w_i, w_j) = \frac{\sum_c mi_{w_i c} \times mi_{w_j c}}{\sqrt{\sum_c mi_{w_i c}^{\ 2} \times \sum_c mi_{w_j c}^{\ 2}}} \qquad (3)$$

Correspondingly, each normalized verb/verb path has a feature vector of normalized noun-subject-normalized noun-object pairs for slots, filled in with mutual information from the actor-object file. Thus, the similarity between one normalized verb/verb path and another can be determined by the same equation.

For higher levels of both ontologies, the same similarity calculation is used. The feature vectors of the higher levels, their "centroids," are simply the average of the feature vector s of their children.

5. Test Run of Indra

In a sample run of one iteration, Indra was run on a small set of 50,000 documents from the Reuters Corpus (of newspaper articles)[7], parsed with the openNLP [8]parts of speech parser, but no entity normalization or coreference resolution was performed. For the standard clustering algorithm, the Weka [9]version of Xmeans was used, so that the number of clusters would be low. Lucene verb [10]stemming and normalization resulted in 1737 active and passive verbs (Actions) for attributes, and 1568 entities (Actors and Objects) for instances in these files. A filter was pre-applied to the entities to keep them in the topic of politics.

Indra divided the entities into twelve clusters based on similar actions. The groupings are entirely automated. Next is a summary of each of the role clusters, of what they appear to mean, and a random sample of up to ten entities from the role cluster. Descriptions of what the clusters mean are guesses, because the clusters are based on the whole set of actions by the entity as computed by the algorithm. Because the clusters are emergent, it will take analysis time to find out why any two entities appear near each other in semantic space.

Cluster 0:

belarussianpresidentalexanderlukashenko

This has a single entity in it. Somehow this man must have done something to single himself out.

Cluster 1:

minister

service

services

office

offices

chairman

ministers

election

president

run

This cluster has very general entities in it, that are related to leadership positions.

Cluster 2:

liberal

vice

presidentbillclinton

democratic

republicans

liberalisation

burundi

runs

reelection

officer

This is a general cluster of the most commonly used words.

Cluster 3:

publicserviceco

foodservice

northwesternpublicserviceco

Service companies.

Cluster 4:

tricomarineservicesinc

affiliatedcomputerservicesinc

simontransportationservices

kllmtransportservicesinc

stewartinformationservicescorp

petroleumgeoservicesasa

southwesternpublicserviceco

atcgroupservicesinc

immigrationministeryuliedelstein

runofmine

Service companies.

Cluster 5:

vicechairman

financeministerjohnfahey

novice

turunsanomat

muenchenerrueckversicherungsag

siteselection

chiefoperatingofficer

communicationministeranwarhossainmanju

interiorministeryogiesuardimemet

mailservice

Cluster 6:

chineseforeignministerqianqichen

presidentlee

chemicalindustryministerguxiulian

primeministertiitvaehi

foreignministerhennadyudovenko

deputyprimeministertansuciller

vicepremier

telebraspresidentfernandoxavier

constructionministerhoujie

defenceministerchihaotian

All of the Chinese officials seem to fall into this cluster.

Cluster 7:

civilsuppliesministerdevendraprasadyadav

ibmcorpchairmanlougerstner

chieffinancialofficerkrischellam

creteilcourtpresidentdominiqueleveque

cochairman

australianwheatboardchairmantrevorflugge

postministerantoniomaccanico

burundiprimeministerpascalfirminndimira

luxembourgprimeministerjeanclaudejuncker

senatedemocraticleadertomdaschle

This appears to be industrial officer types.

Cluster 8:

presidentcostisstephanopoulos

economicsministerguenterrexrodt

lithuanianpresidentalgirdasbrazauskas

presidenteduardofrei

primeminister

vicepresident

pakistaniinteriorministernaseerullahbabar

foreignministerteodormelescanu

presidenthosnimubarak

deputyfinanceministernickoschrystodoulakis

Many eastern European leaders fall in this cluster

Cluster 9:

klmroyaldutchairlinesnv

brunt

gruntal

Cluster of three entities that were mistakes by the parser: an example of how Indra can be used to point out parser errors.

Cluster 10:

nationalserviceindustriesinc

youthservicesinternationalinc

presidentaliabdullahsaleh

itteducationalservicesinc

unitedwisconsinservicesinc

economyministerroquefernndez

paccarincvicechairmanmiketembreull

coministers

usofficeproductsco

cccinformationservicesgroupinc

This cluster has service industry entities. Note that although many of the names contain the word "service," no part of their names were used to group the entities.

Cluster 11:

energyministerjesusreyesheroles

chairmanallenlloyd

chairmanrichardireland

singaporeprimeministergohchoktong

federaltransportministerjohnsharp

nicaraguanforeignministerernestoleal

ukrainianpresidentleonidkuchma

slovakprimeministervladimirmeciar

primeministerpaavolipponen

polishprimeministerwlodzimierzcimoszewicz

This cluster contains many government leaders.

Cluster 12:

presidents

nationalhealthservice

fridayvicechairmanrobertpicow

financeministerbozoprka

uspoliticalcorrespondentpresidentbillclinton

conservativeprimeministerjosemariaaznar

primeministernicolaevacaroiu

presidentclinton

irishrepublicans

austrianpresidentthomasklestil

This is a large cluster containing two thirds of all the entities.

6. Conclusion

More runs are needed on Indra to demonstrate the flexibility of the clusters and thus their suitability for combining the hypothesis-driven ontologies of simulations with the emergent freetext ontologies. Preliminary results show that the data-driven clustering capability works at a basic level, but the open, incremental design makes it a good choice for hybrid combinations of simulation ontologies and data ontologies as compared to LSA and other natural language clustering programs. The difference with Indra is that it approaches the problem of data availability in IW simulations with the philosophy of making the data available dynamically rather than forcing it to match a standard beforehand.

8. References

[1] Landauer, Tom et al, eds. *Handbook of Latent Semantic Analysis* New York: Psychology Press, 2007.

[2] Cimiano, Philipp. *Ontology Learning and Populaton from Text: Algorithms, Evaluation, and Applications*. New York: Springer, 2006.

[3] Lin, Dekang. "An Information Theoretic Definition of Similarity" in *Proceedings of the Fifteenth Annual Conference on Machine Learning*. San Francisco: Morgan Kaufman, 1998.

[4] Ross, Mike and Deborah Duong. *"MICCE: Mutual Information Contextual Concept Extractor"* AGI Workshop, 2006.

[5] Klinov, Pavel and Bijan Parsia. *Pronto: Probabilistic Ontological Modeling in the Semantic Web.* CEUR Workshop, 2008.

[6] Pantel, Patrick and Dekang Lin. "Discovering Word Senses from Text. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining* 2002.

[7] Reuters Corpus, Volume 1, English language, 1996-08-20 to 1997-08-19 (Release date 2000-11-03, Format version 1, correction level 0).

[8] Open NLP, http://opennlp.org/, Feb 2006.

[9] Weka, http://www.cs.waikato.ac.nz/ml/weka/, Feb 2010.

[10] Lucene, http://lucene.apache.org, Feb 2010.

Acknowledgements

Opinions expressed in this paper are those of the authors and do not necessarily represent the position of Augustine Consulting, Inc., Virginia Tech, TRAC-Monterey, or SISO.

Authors' Biographies

DEBORAH DUONG is a Senior OR Analyst at Augustine Consulting, Inc. on contract at US Army TRAC-Monterey. Dr. Duong has nearly twenty years experience in Computational Social Science, including invention of the first agent based cognitive social model in 1991. Her experience includes OSD HBR modeling on the Joint Warfare Simulation (JWARS) project and for the IW team, and US Army INSCOM work on data representation. She authored the Nexus, Oz and Indra programs, and is now working on US Army TRAC's Cultural Geography model.

NICHOLAS STONE is deputy director for Virginia Tech's National Capital Region (NCR) operations. In addition to supporting the executive director's efforts, he works on technology initiatives and programs in the region. Dr. Stone also envisioned and created the Agricultural and Natural Resources Information Technology program at Virginia Tech, and has been involved in many IT-related initiatives in natural resources, engineering and business. He has authored more than 80 scholarly publications and co-authored (with Richard Plant) the book, Knowledge-Based Systems In Agriculture.

BEN GOERTZEL is CEO and Chief Scientist of Novamente LLC. Dr. Goertzel is author of eight books in Artificial Intellegence and Cognitive Science, Editor or four books and author on over eighty scholarly publications. He organized the Artificial General Intelligence conference series and multiple workshops on AI, linguistics, and bioinformatics. He created the OpenCog open source cognitive architecture.

JIM VENUTO is Senior Technical Consultant at Jenzabar. He was formerly a Systems Administrator for Virginia Tech National Capital Region, and System Analyst at ISIS Labs, Inc.