Parserless Extraction; Using a Multidimensional Transient State Vector Machine

Michael Sorah
Rosoka Software, Inc., 950 Herndon Parkway, Suite 280, Herndon, VA 20170

1 Introduction

Rosoka uses a novel approach to extracting information from text, which is not based on the classic mathematical model of a sender and receiver of information but instead of upon a mathematical model of a "third party listener" trying to interpret the information sent between the sender and receiver. This mathematical model predicts the presence of additional information that exists between the sender and receiver which is shared, i.e., "shared context". This shared context model for interpreting possible meaning results in the same mathematical structure as seen in quantum mathematics in that possible messages are distinct from the message meaning. The shared context consists of a number of factors including lexical understanding, linguistic rules, and information or experience that is shared between the sender and receiver, either through world knowledge, personal experiences or prior agreements.

2 Definition of a Parser

A grammatical parser analyzes the syntactic and semantic structure of a sentence, identifying information such as the subject and object of the verb as well as what words group together as phrases or dependencies (see Dowty, Karttunen & Zwicky, 2005; Chen and Manning, 2014; MacCartney and Manning, 2006). In computational linguistics this formal representation can be useful to create meaning from lexical units. In essence, the parser provides a standardized reference for tokens against which a collection of rules can be applied. Although, as discussed below, this is not necessary for successful entity extraction and information understanding.

3 Information Theory, Shared Context, and "Third Party Listener"

Information Theory originated with Claude Shannon's seminal work, "A Mathematical Theory of Communication" (Shannon & Weaver, 1949) to address the problem of information transmission over a noisy channel, such as a radio transmission, from an engineering perspective. Two fundamental tenets of the theory are the **source coding theorem**, which establishes that, on average, the number of *bits* needed to represent the result of an uncertain event is given by its **entropy**; and the **noisy-channel coding theorem**, which states that *reliable* communication is possible over *noisy* channels provided that the rate of communication is below a certain threshold called the **channel capacity**. The semiotics and semantics of human communication are often modeled by looking just at the signal itself; however, it is possible to apply the main concepts of information theory to the problem with effective results. The tendency for very common words to have more entropy than less common words is a robust finding across languages and this word frequency distribution can be modeled by a Zipf distribution. (Zipf, 1932)

Shared Context is information or knowledge that is shared between a sender and at least one receiver. At one extreme, two individuals that speak different languages have very little shared context. They could point, draw pictures, play music, or use other semiotic cues to establish some shared context to expand their communicative understanding. At another extreme, two highly trained professionals

have a very specialized shared context. Their shared context is composed of formal training and learned technical terminology with very precise meanings, along with a world view shaped by some amount of reading overlap. Even though these two highly trained professionals may never have met, they could still have a meaningful technical conversation about their specialized topic of interest. Similarly, the shared context between a group of close friends likely includes mutual experiences and deictic references that preclude understanding for those that are not part of the group. The group could be as small as two people or as large as a community. Criminal argots and L33t-speak represent shared context for particular communities, with in-terms like POS (parent over shoulder rather than part of speech), and LOL (laugh out loud).

Rather than the classic model of Message Sender and Message Receiver, (Shannon & Weaver, 1949; Schiffrin, 1989; Tannen, 1993; Schiffrin, Tannen & Hamilton, 2008) Rosoka's algorithms are based on the model of Message Sender, Message Receiver, and Third Party Listener. In this model the Third Party Listener is constructing a model to interpret the information that was sent to the user. In the construction of this model, the Third Party Listener is not privy to the shared context of the Message Sender and Message Receiver, and instead has to construct possible interpretations that will likely lead to understanding the message. The information available in the shared context is assumed by the Message Sender to be present already in the discourse, and consequently it is not transmitted in the singal. Without the non-transmitted information, there exist multiple possible interpretations of the signal and it may be misinterpreted by the Message Receiver. This assumption predicts that language communication can be modeled using non-Kolmogorov probability theory, as argued by Aertz et al. (Aerts, Czachor and D'Hooghe n.d.). It also follows that the processing rules for such should also have a quantum structure, i.e., the distribution of rules that the sender and receiver use should follow the Zipf distribution so as not to require near infinite energy. The implementation of Rosoka's algorithms results in such a rule distribution, as discussed later in this paper.

4 Semantic Vectors

To construct a third party listener model, Rosoka uses a vector space of possible interpretations of a stream of tokens. The vector on a token in Rosoka is called a "semantic vector," or SV. This allows the tool to create a vector space of state possibilities for each token or set of tokens in a token stream. In this vector space, a token can have multiple possible meanings at any step in the process, and the meaning(s) can change during the processing.

The vector space in Rosoka is finite in that it is predictable and repeatable; the vector space itself is defined for each position on the length of the vector at runtime. The semantic vectors, on the other hand, are non-finite because they represent a set of possible states that change during processing based on the surrounding vectors. Thus Rosoka is a cross between a finite state machine and a non-finite state machine. Rules in Rosoka operate on the semantic vectors to enable or disable particular portions of the vector space based on an individual token's vectors and surrounding vectors. The vector space for a token is represented by the possible states "is," "isnot," "true," and "false, " allowing the possibility for a vector to be "is" and "true," or "is" and "false," or "isnot" and "true," or "isnot" and "false." This quantum structure represents the intended meaning of the sender (is/isnot) and interpretation of receiver (the true/false). The vector space can also be modified *post hoc* through the process of recursion, based on changes to a particular vector or its surrounding vectors.

As a real world example, consider a reader encountering a novel word. A dictionary definition will indicate multiple possible meanings of the same word. To resolve the meaning, the reader needs to decide which definition is most likely based on the context of its usage. For example, the word "can" may mean: (noun) a type of container, (verb) indicating ability, (verb) to fire, (verb) modal, (noun) abbreviation for Canada, (Navy slang) a destroyer, (common typo) other words with near spellings such as "cane" or "scan". Using the available contextual information, as well as shared context with the sender, the reader must determine the intended meaning of the signal.

5 Multidimensional Transient State Vector Machines

The Rosoka process can be thought of as a multidimensional transient state vector machine, because the semantic vectors are changing with the state of the processing based on the surrounding vectors. The initial state is established through a lexical lookup for each token or contiguous sets of tokens to see if there is are semantic vectors defined in its lexicon. The token sequences are then compared to the rule set to find the first applicable rule. If a rule matches, it is then applied to the token sequence. The rule may change the vector state on the token, or it may combine sets of tokens to make a new token with a new vector space.

For example, consider this article's authorship line as a token stream of 3 tokens: "By," "Michael," and "Sorah."
The lexicon would set the SV space as follows (only true values are displayed):

```
<lex><word>by</word><sv><adverb/><prep
/><locative_prep/></sv></lex>
<lex><word>michael</word><sv><given_na
me/><given_name_male/><sur_name/></sv
></lex>
```

and "Sorah" is not know to the lexicon so it would be assigned.

```
<lex><word>sorah</word><sv><unknown</
sv></lex>
```

In Rosoka, it is possible to write a rule that will tell the engine that when there is a preposition followed by a given name and then by a unknown word, combine the given name and unknown word to a new token, set the vector to a person, and turn off the other vectors states for the prepositional phrase. After processing, the token stream would now be represented by:

```
<lex><word>by</word><sv>prep/></sv>
</lex>
<lex><word>Michael
Sorah</word><sv><PERSON/></sv></lex>
```

In stark contrast to a parser, not only have we modified the vector on the tokens but we have also modified the token stream. From this processing state, rules that would apply to unknown words, surnames, or given names would no longer be applicable, and not need to be checked. In the implementation of this parserless construct, our dictionary not only includes typical part of speech tags, but also includes pragmatic tags (e.g., given_name) that allows for processing short cuts, bypassing many processing states and thus taking less computational energy. For example, there is no need to define Michael as a noun, much less a proper noun, because it is not relevant to the information content.

Rosoka also contrasts with a classic rules engine because Rosoka's semantic vector space allows multiple conditions on the vector to be simultaneously checked. For instance, it is not necessary to check every possible condition for finding a person name; once a rule has matched, the vector space changes and makes additional checks unnecessary. Thus the equivalent of thousands of classic rule conditionals can be collapsed into a single vector space rule, which requires less entropy to process. This

provides a degree of fitness measure for efficient rules measured against information value. The significance is that only a small number of rules are needed. Rosoka has hundreds of rules; traditional pattern based tools have tens of thousands of rules to accomplish the same tasks.

A subtle, yet important difference between Rosoka and classic binary rule logic is that the algorithms check to see if a rule can match the vector space sequence rather than if a rule is activated or not. If the pattern should apply, it will; otherwise, it will not. In effect by looking at a single bit, the process does not even need to check any rules that cannot apply to that bit. This may be a subtle distinction from classic binary rule logic, but it provides a savings of orders of magnitude in terms of computational throughput speed.

Some systems use a rule precedence fall-through methodology; in these systems, rule order is paramount to processing success. Under such systems, adding additional rules means that the entire rule order chain needs to be re-evaluated to prevent entire logic branches from unintentionally being ignored. These systems add a linear-to-exponential amount of processing computation for each new rule added. This is avoided entirely with Rosoka's methodology.

6 Types of Knowledge and Machine Learning

Human language learners typically demonstrate three types of knowledge: rote knowledge, compositional knowledge, and dynamic knowledge. Since the Rosoka extraction engine can change state and alter the token stream, it is able to leverage these three types of knowledge.

Rote knowledge is the knowledge that is inscribed in the lexical lookup tables. This is represented by the values associated with each token or set of tokens captured in the Rosoka lexicons.

Compositional knowledge is the knowledge encoded in localized canonical rules used to interpret the meaning of a token or collection of tokens. An entity like *John Smith* can be recognized as an person because of the component pattern of *given name* plus *surname*; John is a known given name and Smith is a known surname. The two tokens together comprise a valid name regardless of whether both names have been encountered together before. Any combination of known given names and known surnames could make a valid match.

Dynamic knowledge is represented by rules that need the larger linguistic context to determine the appropriate interpretation. In a sentence like "Chinua Achebe is a Nigerian author," the name *Chinua Achebe* is easily recognized as a person because of the linguistic context—authors tend to be people. Even if the tokens *Chinua* and *Achebe* are unknown in the lexicon, Rosoka is able to extract the entity using the context. Once *Chinua Achebe* is recognized as a novel name, it can be extracted in other less semantically rich contexts.

Again we see the **entropy effect**; rote information takes less computational energy, while dynamic information takes more. Converting dynamic discovery to rote information is based on the balance between the degree of fitness for computational efficiency and more complex rules used to recognize the relatively rare occurrence of high value information. The entropy in this case is based on the amount of computational effort to deal with false positives and the consequences of missed information. In Rosoka, once *Chinua Achebe* is identified as a person name, it can be recognized using a very inexpensive rote rule, and more complex, costly rules need not match. Because the name is now lexicalized, Rosoka can skip using a rule to find the name based on its components as

well as a far more expensive rule that uses the sentential context.

The Rosoka engine allows users to vet values that are discovered dynamically to be either incorporated into the lexicon or "unlearned" as a *not* statement, e.g., "not surname." This vetting feature is important because for very large sets of documents (i.e., millions) we see the phenomena of false positive creep, or reduction in precision, that occurs when statistical learning systems process large data sets. Additionally, Rosoka eliminates the need to hand tag large sets of training data, because Rosoka is effectively self-tagging.

7 Pattern Resemblance to Classic Linguistic Rules

Rosoka rules in many ways resemble classic linguistic rules, in part because they parallel the way humans understand and interpret language. However, there are some important distinctions. Rosoka rules include instructions regarding how many tokens to combine and which semantic vectors to set or unset when a rule matches. Additionally, the rule specifies attributes to track with the tokens and the *is* and *isnot* conditions for the token stream. An individual token's position in the token stream is expressed as a relative offset. Figure 1 shows a rule for identifying three part names as a person entity. While the rules may appear to be Boolean in nature, they are actually expressed as a quad state of is/isnot and true/false, with *true* indicating that the vector positional name is present in the <sv> tag and *false* indicating that it is not. Rosoka's rule syntax allows a rule writer to think in terms of the Boolean equivalent, similar to a Newton approximation for general relativity, with logical AND and OR. The AND condition is akin to having multiple conditions for a token, and the OR condition is akin to having multiple items in the <sv> list.

```
<Rule ID="person_cf-1115a">
  <description>three part names e.g. John Foster Wallace</description>
  <order>0</order>
  <result>
    <combine>2</combine>
    <sv><PERSON/></sv>
    <nolonger>
      <CONVEYANCE/><vehicle/><placename/><given_name/>
      <month_name/><CONVEYANCE/><vehicle/><sur_name/>
      <generic_person/><generic_org/><sur_name_arab/>
      <given_name_female/>
    </nolonger>
    <attributes>
      <given_name><T offset="0"/></given_name><sur_name><T offset="2"/></sur_name>
    </attributes>
  </result>
  <when>
    <T offset="0">
      <IS><sv><given_name/></sv></IS>
      <ISNOT><sv><title_pre/><noun/><verb/></sv></ISNOT>
    </T>
    <T offset="0">
      <IS><sv><cap_word/></sv></IS>
    </T>
    <T offset="1">
      <IS><sv><given_name/><sur_name/></sv></IS>
    </T>
    <T offset="1">
      <IS><sv><cap_word/></sv></IS>
    </T>
    <T offset="2">
      <IS><sv><sur_name/></sv></IS>
      <ISNOT><sv><modal/><verb/><noun/><month_name/></sv></ISNOT>
    </T>
    <T offset="2">
      <IS><sv><cap_word/></sv></IS>
    </T>
  </when>
</Rule>
```

Figure 1: Example Rosoka rule for extracting three part person names.

8 Multilingual Processing

Since Rosoka is not tied to a parser but instead to a vector space, Rosoka rules transcend the language that the token is written in. To process in a different language, Rosoka requires only the lexical mapping to the semantic vector space. So, if a Korean document contains “블라디미르 푸틴,” it will have the same semantic vector as *Vladimir Putin*. Or "國務院" has the same vector as *Department of State* and well as the transliteration of *guo wu yuan*. This means that the engine itself doesn’t care what language the tokens stream is in, only the word sense order. Complementary distribution results in rules for one word sense order not matching rules when they are not in that word order (i.e the rules won’t get applied). Thus the engine can process in any language, without requiring an intermediate translation, and the accuracy, or precision and recall, are only dependent on the breadth of the lexical entries for that language.

9 Rosoka Rules and the Zipf Distribution

Similar to the way that lexical item frequency follows a Zipf-like power law distribution, the mathematical model of the semantic vector space predicts that rules based on such a vector space will also follow a Zipf distribution. In practice, the rule distribution of matching rule patterns in Rosoka follows the predicted Zipf distribution. Figure 2 shows the rule set matching against a generic corpus of documents, clearly illustrating the classic Zipf distribution.

Rule Distribution

Total Rules: 676 Total number of Rule matches: 19050

Sequence	Rule ID	Description	Fired	Zipf frequency	Distro
647	GRundo_place	Tidy up Place	2777	100.00000	
200	GRtidy_nl-0002	Numbers are no longer unknownwords	1640	59.05654	
40	MT1	URI	1368	49.26179	
46	MT1112	plain signed decimal numbers to indicate structure	808	29.09615	
646	GRundo_org	Tidy up Organization	721	25.96327	
323	GRorg_nc-7028	orgname with no context e.g., Oxford Research Group	624	22.47029	
34	MT1979	An ID Number of some sort	600	21.60605	
35	MT19790	multipart international phone numbers with optional country code	596	21.46201	
609	GRgeneric_nc-0002	non-contextual generic organizations e.g. group	563	20.27368	
217	GRperson_nl-1106	Provide knowledge that this is not a possible sur name	515	18.54519	
52	MT1979003	Finds possible idnum with multiple numbers attached by punctuation	512	18.43716	
444	GRphone_nl-0098	PHONE NUMBER Cleanup to keep other things from grabbing it	450	16.20454	
440	GRphone_lc-0005	Phone Numbers	405	14.58408	
441	GRphone_lc-0006	telephone rules with preceeding context	366	13.17969	
644	GRplace_rr-9001	City, place, and province names with no other context e.g., Bali	363	13.07166	
293	GRplace_nc-0012	catch-all nation names alone e.g., Australia	357	12.85560	
608	GRgeneric_nc-0001	non-contextual generic person e.g. mastermind	317	11.41520	
214	GRperson_nl-1104a	Rules out possible surnames if they are preceeded by a definite article e.g. the judge	316	11.37919	
517	GRevent_nc-0001	non-contextual events, both named and generic e.g., sabotage	315	11.34318	
284	GRplace_nc-0003	placename e.g., Sulu Archipelago	272	9.79474	
161	IT2	This handles 1900-2099 for stray years- -GLK	260	9.36262	
44	MT21	Finds US phone numbers with area codes in additional forms	235	8.46237	
585	GRpredicate_nc-0001	predicate rule for job titles	203	7.31005	
201	GRpro_nl-0001	pronouns that are also asian surnames cannot be with verbs e.g. he knows	195	7.02197	
237	GRperson_cf-1105b	Rule to find simple Given Name Sur Name e.g. Ingrid Martonova	192	6.91394	
292	GRplace_lc-0011	nation names with a prep e.g., in Indonesia	182	6.55384	
176	IT1972	Military time including H for hour marker	179	6.44580	
177	IT1972a	Military time for hours minutes	179	6.44580	
223	GRperson_nl-1141	Rule to eliminate false Sur Names e.g. in August	138	4.96939	
38	MT13a	Finds capitalized initials with a period	124	4.46525	
199	GRtidy_html-0001	Html entity markup tags should not be NUMBERS or IDNUMS	101	3.63702	
434	GRts_rr-0053	Dangly day names (all cap strings and langs without caps)	93	3.34894	
280	GRperson_md-1132	Set PERSON attributes to female if female name e.g. Julia	90	3.24091	
420	GRts_rr-0044	Relative: Year number by itself 2002 ::RECALL HEAVY::	88	3.16889	

Figure 2: Frequency distribution of rule matches

Because of the tendency for vector-based rules to follow the Zipf distribution, a small number of rules can provide a very high level of comprehension. Unlike other NLP tools that use parser-based rule systems, Rosoka can successfully extract entities using just a few hundred rules. Rules beyond the basic out of the box capability become either exception handling or domain-specific pattern recognition, and these can be implemented and tested very quickly. As Figure 2 shows, additional rule writing rapidly approaches a point of diminishing marginal returns.

By contrast, training a statistically-based learning machine on these high value but infrequent patterns requires providing a statistically significant number of examples, which, given the inherent infrequency of such information, represents a significant level of effort.

Rosoka's extraction engine leverages important aspects of communication theory, quantum vector space of state possibilities, and the Zipf distribution of lexical and linguistic pattern frequency to provide a uniquely efficient and effective method of entity extraction. Rosoka's algorithms allow for multiple possible meanings throughout processing, recursive pattern matching, and the addition of domain-specific rules with negligible additional processing cost.

References

Aerts, D., Czachor, M., & D'Hooghe, B. (2006). Towards a quantum evolutionary sheme: Violating Bell's inequalities in language. In W. Abraham, & M. Noonan (Eds.), *Evolutionary Epistemology, Language and Culture.* . Amsterdam: Springer Netherlands.

Chen, D., & Manning, C. D. (2014). A Fast and Accurate Dependency Parser using Neural Networks. *Proceedings of EMNLP*.

Dowty, D. R., Karttunen, L., & Zwicky, A. M. (2005). Natural language parsing: Psychological, computational, and theoretical perspectives. *Cambridge University Press*.

Mandelbrot, B. (1965). Information Theory and Psycholinguistics. (B. W. E.Nagel, Ed.) *Scientific Psychology*.

Marneffe, M.-C. d., MacCarney, B., & Manning, C. A. (2006). Generating Typed Depndency Parses from Phrase Structure Parses. *In LREC*.

Powers, D. M. (1998). Applications and explanations of Zipf's law. *Association for Computational Linguistics: 151-160*.

Schiffrin, D. (1989). Conversation analysis. In *Linguistics: The Cambridge Survey: Volume 4, Language: The Socio-Cultural Context* (Vol. 4).

Schiffrin, D., Tannen, D., & Hamilton, H. E. (Eds.). (2008). *The handbook of discourse analysis.* John Wiley & Sons.

Shannon, C. E., & Weaver, W. (1949). *The Mathematical Theory of Communication.* University of Illnois Press.

Tannen, D. (1993). *Framing in discourse.* Oxford University Press.

Zipf, G. K. (1932). Selected Studies of the Principle of Relative Frequency in Language. *Cambridge, MA: Harvard University Press*.