Speak\*Morph: A Graph Completion Algorithm

Drexel University

Walter Gress

Advanced Artificial Intelligence

Revision 1.0

February 15, 2013

**TABLE OF CONTENTS**

Algorithm Synopsis (Topic) 3

Propositions and Theorems 3

Graph Structure 7

Reduction and Query Reformulation 26

Graph Mapping (Matching) 29

Combination of language graphs to generate Rosetta Graph R 37

Machine Learning: Training Rosetta Graph R via Dictionary Data 38

Architecture of Application 39

References 45

# Algorithm Synopsis (Topic)

There exists, given a set of graphs representing known languages A, and a set of graphs representing unknown language B where a is an element of A and b is and element of B, a way to generate and maintain graphs of both A and B. There exists a method through context comparison of a of A, rules such as Verner's Law and Grimm's Law, graph matching algorithms, comparison from the set of sentences over the graph, and other qualities inherent in the graph being used. In order to map between the set of complete languages for all a of A to an incomplete language b of B, where this mapping may be used to replace, insert, delete, or modify the nodes and edges of graph b in such a way that the language b is no longer classifiable as of the set B and may be considered a fully translated language of the set A of translated languages.

# Propositions and Theorems

It is of my belief that should the following four propositions hold as true (or very close to true by approximation), an approximate, universal Rosetta stone like data structure could be constructed. It would be a language interpreter that not only understands present languages, but also new, incomplete, languages should the system be presented with them in the future.

Although the first proposition presented is at present simply conjecture and difficult to prove without advancements in neuroscience, should it be proven to be true or approximately true (true to a point) then it would serve as a steady-enough foundation for the propositions that follow from it.

Please note that this project will not be an effort to prove the theorem directly and deduce a graph structure and matching algorithm from the following propositions and conjectures. Primarily because the base propositions cannot be proven using our modern day tools, some are left as conjectures and to be treated as self-evident and without proof. These propositions are primarily, but not limited to, the first proposition, and also the fourth.

In order to establish the successfulness of this system we must assume the base propositions of the theorem true and attempt to build a trial application adhering to the requirements given by these propositions, in order to show that the propositions presented, empirically hold true.

**Proposition 1 – Similarity between Human Brain Structures**

Given any two human brains, there exists a structural similarity, that when two are presented with their structural differences removed from comparison data, any two human minds are structurally similar up to a point given by the constant sk ,where sk is an inverse threshold constant. When sk is at its lowest (0), there is absolute similarity between any given two brains, that is, they are structurally identical such as in newborn twins. The value of sk is unbounded in the positive x direction, meaning that any two brains may be structurally different in many different, uncountable, ways.

**Proposition 2 – Representation of the Human Brain by Directed Graph**

Given any human brain, its structural data can be represented in the form of a specific directed graph data structure. Lee Osterhout et. al in [3] suggests indirectly suggests this, saying “Neural network simulations provide a convenient framework for understanding effects of L1 usage on L2 learning….” (page 520)

**Proposition 3 – Threshold Isomorphism of two Human Brains**

Given Proposition 1 is true, that two brains are structurally similar up to the threshold point sk, and given Proposition 2 that every brain’s structural data can be represented in the form of a directed graph data structure, this third proposition states that there must exist an isomorphism up to threshold tk between the two graphs representing the two brains, where the threshold tk represents the structural threshold sk mapped to represent thresholds in the graphs of the brains instead of the direct brain structures.

Assuming that the above three propositions hold true, with the following proposition (4) holding true, there is proof that we are able, should we choose to, develop a system that will take a database of given languages and use that database to solve any non fully-translated language into a translated one via the properties of the graph, and isomorphism between the two graphs using a particular, yet to be established, graph matching algorithm.

**Proposition 4** **The Language Sub Graph Proposition**

Given any Human Brain there is a specific structural portion devoted to language (Wernicke’s and Broca’s areas to be specific). This is a well known fact with two specific areas notably designated in the brain as handling speech and language (Wernicke’s and Broca’s areas respectively). (See Shara’s article [6]– page 6, page 9, page 10, ) Therefore, it can be surmised that given any graph of the human brain, there exists a portion or **sub graph** of that graph that is devoted entirely to language. Therefore, it can be shown, that if the prior three propositions hold, then proposition 4: That there is a language sub graph in every human brain, holds.

**Theorem of Isomorphic Language**

Given all the propositions above we arrive at this theorem. Given any two average human brains, there exist two graphs isomorphic to each other to the threshold measurement tk, such that there also must exist two language sub graphs in each brain, isomorphic to each other up to the threshold measurement tk.

Note that the brains in comparison must be of average type, brains that

deviate strongly from the average, such as those with head trauma or those that suffer from certain illnesses, or of even extreme IQs, may have nearly altogether different structures, therefore passing the threshold measurement tk.

If we are to encode every language into a graph structure where all languages were isomorphic to threshold tk, **we could use this isomorphic property of the graphs to fill in missing nodes and edges in the graph** of a language that has not been fully translated, or is not translated all through a graph matching algorithm. We could use this system, as a Rosetta stone, since the new language’s graph, once generated, **according to the above propositions**, should be isomorphic to known languages, with correspondences existing by the very nature of the language graph.

**Supporting Information**

Zhou and Yu in their paper “Graph-based Language Model of Long-Distance Dependency) [2] review the idea of a semantic space (semantic web (pgs 38 and 39) showing that a text document of a given language can be represented as a weighted digraph. We expand upon this idea, considering that since there exists a text document as a subset of a language, if the text document can be converted into a weighted digraph, then the language itself should be able to be converted into a weighted digraph as well (even if only as a collection of connected digraphs generated from individual text documents).

Pawarand Zaveri in “Graph Based Pattern Matching” ,[1] pg. 1023 describe Graphs, Graph Isomorphism, SubGraphs, and Graph Matching.

Pawarand Zaveri in “Graph Based Pattern Matching” , [1] pg. 1024 describes graph matching process in detail. (By definition). Pawar and Zaveri use a fitness function to measure the similarity between matched vertices and edges.

**Theorem of Isomorphic Graph Mapping Corollary**

Given two isomorphic graphs representing two distinct languages, S1 and S2, it is given that the two graphs may be isomorphic, but that because the graph matching problem is NP Complete (or hard?), an optimal solution is not necessarily solvable in real time, only an approximation.

However, for certain languages S1 and S2, there already exist certain laws and heuristics for translating between grammars such as Grimm’s law and Verner’s law, among others. In addition, when performing a mapping between two graphs where one graph is complete and the other is incomplete, existing node content can be used to complete the incomplete graph by matching existing node content with node content in the complete graph. These rules, along with training data, can provide a mapping between the two given languages without the requirement of graph matching based on structure alone. However, if a correspondence can be found between the properties of the two languages being mapped to each other and the properties of a graph matching algorithm that exists or doesn’t exist, then that mapping should hold over known languages S and any incomplete language I. **To speak generally, the rule(s) for language translation should be able to be mapped to the rule for this instance of graph matching.** That is, a graph-matching algorithm may consist of an abstraction of the general rule(s) of translation when the languages are abstracted by graph structures.

# Graph Structure

Justification for the usage if Graphs to represent Language

According to Sowa [11], conceptual graphs have been used to represent language for a long period of time, the earliest forms known as *existential graphs*, invented by the philosopher Charles Sanders Peirce. The earliest forms on a computer were the *correlational nets* by Ceccato and the *semantic nets* by Masterman.

**How to structure the graph**

Conceptual graphs to represent language according to Sowa [11] have several different themes common to all graphs representing language:

1. Concepts: Nodes of the graphs represent concepts of entities, attributes, events, and states. In our graph nodes are representative of adjectives, nouns, and verbs acting on nouns with adjective descriptors attached.
2. Instances: Different nodes of the same concept type refer to different instances of that type, unless they are marked with a *name* or other symbol to indicate the same instance. In our graph instances are marked using existential and universal quantifiers, and are unified if possible, with the nodes acting as their own instances.
3. Conceptual Relations: Arcs of the graphs represent relationships that hold between the instances of the concepts they are linked to. Labels on the arcs indicate case relations as well as causal and logical links between propositions. In our case we use directed, property-less edges to represent relationships between nouns and verbs (which are divided into bipartite graph). Verbs are represented as nodes between nouns in trigrams and therefore the causal and logical information is stored within the verb nodes embedded in the trigram.
4. Type Heierarchy: Concepts are ordered according to generality. This is accomplished in our graph by the logical and set connectors that indicate subsets, existence, is-a and has-a relationships.
5. Inheritance: This is handled in the same way 4 is handled.

The beginning question is we should structure the graphs that encode the syntax and semantics of the language.

Given a dictionary of a language, a graph can be constructed including both quantifiers (to divide between universal predicates and predicates referring to individuals) borrowed from predicate logic, as well as “threshold feed-forward constants” similarly borrowed from Bayesian Networks and Artificial Neural Networks in that they mark a value between 0 and 1 upon an edge indicating whether transition may occur from node to node along the marked edge. This feed-forward nature is similar to that of ANNs.

Because of complexity issues, given a document and a plethora of sentences, it may not be represented as a simple Bayesian Network or Semantic Network, but its structure represented similar to a Bayesian Network or Semantic Network and its sentences stored in another set, with reference to locations (nodes and edges) in the network for each sentence.

Thus we define an arbitrary directed graph G=(V,E) and its set of sentences over the graph G as C = (p1, p2,…., pn), that is, the set C represents a sequence of paths through graph G. Each path corresponds semantically as a sentence in the graph, while the graph itself represents the syntactic structure of the language. This is also similarly structured in the brain, that is, there is a spatial distinction between parts of the brain storing syntax and those storing semantics ( [6]shara’s paper, page 20). This is also said in [3], page 510, part 2.

**Sentences as Trigrams**

The sentences of the input Dictionary shall be divided into trigrams for more efficient use. By dividing the sentences into trigrams, common relationships between certain common words will be strengthened, where as if each sentence was treated as an entire entity of more than one node, the graph structure would not be as compact (number of connections per node would be low) and achieving a knowledge base would take more time since there would be a high number of sentences but a low number of correspondences between sentences.

**Utterance Paths (Sentences) Over Syntactic Graph**

Similar to Sowa [11], the graphs designed for Speech\*Morph do not move large amounts of data around but maintain a set of sentences, similar to how APSG in [11] maintains utterance paths. This saves in both resources and time of execution.

Graph

**Feed-forward Threshold Variables in the Network**

As for the feed-forward thresholds, each feed-forward threshold starts at a very low number given by constant th, very close to 0 (but never zero itself). As the threshold passes transitions through its edge, it is understood to be increasing the frequency and therefore commonality of this node-to-node relationship. The feed-forward threshold increases, approaching but never reaching 1 (exact truth or exact certainty). The feed-forward thresholds are asymptotic, that is, 0 < x < 1, where x is the threshold value.

We can calculate the way words occur together as follows:

P(Trigram) =

Thus the probability (feed-forward threshold) of a trigram in the network is equal to the number of similar trigrams divided by the number of all trigrams in the network.

Probability is assigned to the threshold weights on the edges of the trigrams, updated using the above formula??

**Feed Forward Thresholds and Sin Curve**

The frequencies used to train the graphs work off of an inverse sin curve, slow at the start when the frequency is 0, quicker in the middle, and then slower again as the number one is approached. This dual asymptotic curve allows for the data to be trained without ever reaching a state of absolute uncertainty (0) or absolute certainty (1).

The curve is displayed below:

1

0

Travelling along (iterating through) values of Y yields X values asymptotic and useful for the frequencies, whose contraints are 0<f<1, for anything frequency f.

**More on the General Graph Structure**

**Use of nodes for verbs**

The graph uses verbs as nodes instead of assigning verbs to edges in order to prevent multiple instances of the same verb occurring – undoing the compactness that is necessary in the graph for tractability. In a graph system where nouns are represented by nodes and edges by verbs, the graph will quickly reach a state where there multiple nodes with identical, individual verb edges, losing the significance of the verb as a part of speech and as a part of the semantics of the network. This graph structure is based off of a simple semantic network. Z. Cai discusses this in [9], detailing a simple use of a semantic network to represent language concepts, where one node would feed forward to activate another node, etc. in the semantic network.

Each edge however also includes a “weight”, the threshold feed-forward constant spoken of above. This weight signifies the “strength” of the relationship between noun and verb and adjective and noun and so forth.

**Feed-forward Obfuscation**

There may occur a problem if only one verb node exists for several sentences. In this case, the original meaning stored in the graph becomes obfuscated. A diagram of such a situation is below:

Bird

Dog

Person

EATS

Bird Food

Dog Food

Human Food

**Fig 1.3**

Bird

Dog

Person

EATS

Bird Food

Dog Food

Human Food

0.9

0.9

0.9

0.9

0.5

0.5

Because of the identical usage of EATS node and the lack of path storage, it seems as though the graph is suggesting that a Person will eat Dog Food or Bird Food, and a Bird might eat Dog Food, etc…. The graph is obfuscated and through the attempt to make it more compact has actually distorted the information stored.

Even if with the given feed-forward probability as below, the problem still exists:

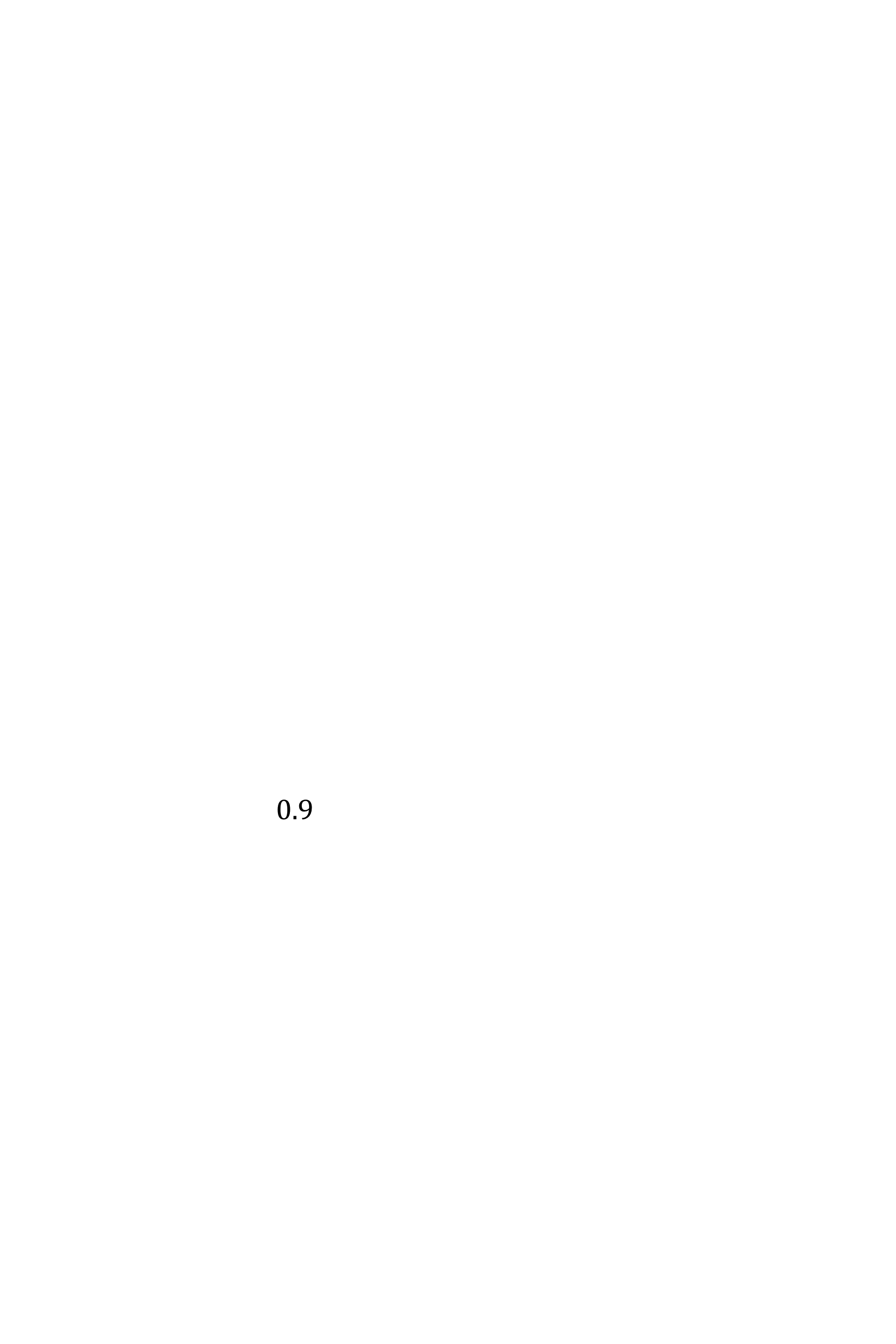


Fig 1.4

Again, the numbers on the edges are not strictly probabilities, but a real number between 0 and 1 exclusive indicating the likelihood a transition function will transition from the current node to the next.

**Bipartite Graphs**

An idea to perhaps improve the performance and structure of this graph is the transition of the above graph into a bipartite graph and another bipartite graph constructing a sentence that is linked together through a transitive property, each graph containing its own part of speech. Bipartite graph matching has been shown to be more easily solvable than traditional graph matching (see Wikipedia articles and Barpartite Matching Articles)

An overview is presented below:

Adjectives

Nouns

Verbs

Fig 1.5

Given a sentence, the nouns would be extracted along with the verbs. The verbs placed in the bottom set, the nouns in the top, and the adjectives at the very top. Then edges would be drawn between the sets indicating the structure of their usage in a sentence.

For example:

***Nouns***

Bird Dog Cat Monkey

Bird Food Cat Food Dog Food Monkey Food

***Verbs***

EATS

Fig 1.6

Again, despite the overall structure is changed into a Bipartite Graph, the overabundance of outgoing nodes and incoming nodes to just one verb, EATS, will result in a still inefficient data structure.

**Solving Singular Instances**

A possible solution, and the solution we use in Speech\*Morph to the singular instances problem above with verbs is to create multiple, linked instances. This can be conducted in a variety of ways. The goal is to shrink and consolidate the graph, such that the graph becomes less complex and less wide but retains its original knowledge base.

The process to do so I have labeled “unification” for the graph.

**The use of quantifiers to handle generics and ambiguous nouns**

Sowa[11] describes the use of generic concepts, objects which contain only type information. These objects could be used in places where the noun is ambiguous, where there is an unspecified individual of that type. This is primarily handled in our system by the use of the “for all” quantifier on a noun node, the use of a so-called “noun-less” node may prove fruitful in further designs. Also in Kamel [12], Kamel’s GKR (Graph Based Knowledge Retrieval System), conceptual graph consists of two elements; “concepts: people’s underlying interpretations of things that are perceived” and “relations: relationships that can occur between concepts.” In Kamel comparison to our system, the concepts correspond to the nodes in the graph and the relations to the edges, purporting a close isomorphism between Kamel’s method and our own.

**Unification Stage 1 : Unifying** Quantifiers

The “for all” quantifier, indicating that the node in question is universal and connecting edges apply to all cases of the node. For example two sentences with the for all qualifier:

*All planes have wings. All planes have engines.*

These two sentences, although both using the noun “planes”, refer to the proper noun, universal version of Plane, that is the case and thus can be combined into one statement:

*All planes have wings and engines.*

However, instead of connecting these two nodes with an edge (with the conjunction quantifier for example), we merge the two nodes. So that the left graph of Figure 1.7 becomes the graph on the right.

Engines

Edges

Wings

Wings

Fig 1.7

Notice that above, the node, (in this case the is implicit),

has been merged together at the “Planes” Node, but at the “have” node (being verbs) have not been not changed structurally. They are handled differently as we will see later.

**Unification Stage 2: Existential Quantifier**

Multiple instances of the **same** non-universal, unique node (such as multiple existences of a node with the are also processed in a unification process. Unified similarly to the universal quantifier, but **ONLY** if it can be shown that the two nodes reference the same node. I.e., there may be three or four or a million cases of  **Dad** nodes in the network, only three or four of these nodes may be eligible for unification. In the case that the three or four are 100% the same “Dad”, then they may be unified, shrinking the graph further and reducing its complexity as well. This process of unification occurs when the graph is being constructed, instead of creating individual **Dad** nodes for each occurrence in the dictionary of **Dad**, context is used to unify them.

**Unification Stage 3: Action Quantifier**

Obviously the network would not function with only noun nodes and predicate logic quantifiers. In addition to the above we also have verb nodes, which are denoted as having a as a quantifier, this being essentially just a notation to differentiate from nouns with universal and existential quantifiers. Nodes with quantifier are linked using a non-directed edge, indicating they are related by not causally related in any form.

At this point we’ve modeled language data so that ideas (nouns, verbs) are interlinked and reusable throughout the network. We would like to use the structure of the network to store our sentences, which if the reader recalls are paths through the network. In order to do so we have created a graph with quantified nodes, weighted **frequency** **feed**-**forward** edges, and an ontology that divides nouns, verbs, and adjectives into bipartite graphs.

Now I will present an example of pre and post unification on the proposed graph structure, showing its usefulness.

EATS

**0.95**

**0.50**

**0.99**

**0.7**

**0.5**

**0.95**

Fig 1.8

Unification of PAST node, no other unification performed.

EATS

**0.95**

**0.50**

**0.99**

**0.7**

**0.5**

**0.9999**

Fig 1.9

The above unification is of the action quantifier . However, instead of merging the nodes, which would result in the obfuscation we had earlier in figures 1.3 and specifically 1.6, we link the verb nodes with an undirected edge. An undirected edge is chosen as undirected partly to make the edge type unique, and also to show that there is no causal relationship between the two nodes. That is, EATS is equivalent to EATS no matter what position EATS occupies in the graph. By traversing the EATS links, we can learn what “EATS’” without having the EATS nodes merged, ruining the syntactic structure of the graph. There is also no feed forward weight, indicating that all EATS occur in equal frequency, essentially, that each EATS node is equal to another.

**Logical Connectives**

It seems prudent to assign certain universal symbols that appear in logic as well as an augment of these symbols, to symbolically represent very common verb relationships between nodes. The encoding of these verbs as symbols may prove useful for using Speech\*Morph for theorem proving in the future. Most of these nodes are borrowed from predicate logic and object oriented principles.

|  |  |
| --- | --- |
| Connective |  |
| = | Is a (bidirectional synonym) |
|  | Is an element of |
|  | Is a subset of |
|  | Is a proper subset of |
|  | Aggregate of (Has a…) |
|  | Directional (If…then) |
|  | Bi-Directional of (if and only if) |
|  | Logical AND |
|  | Logical OR |
|  | Logical XOR |
|  | Standard Verb (Directed Edge….runs, jumps, etc.) |
| ---- | Standard Logical Connection (Verb with no additional info)  (Without causal information) |

Fig 1.10

Similar logical connectives, such as “is-a” and “has-a” are used in the construction of the directed acyclic graph structure (DAG) used in [16] for the pattern matching over DAGs and have been added to the standard list of logical connectives.

**Another Illustration of Unification, this time with logical connectives:**

**Fig 1.11** (Ununified Graph Using Logical Connectors)

The logical connectors between Dad and Truck and Dad and car and the other similar ones are the “has-a” connectors, not subset connectors.

**Fig 1.12 (Completely Unified Graph with connectors)**

In the above figure, the two Existential Dad nodes have been merged. Also, as is evident, the verb operators (the aggregate operator ) and the subset operator () are unified by being linked with an undirected edge by the unification stage 3.

**Unified Graph with Cyclic Connectors**

The final addition to our graph structure is the additional quantifier/connector represented by an additional node linking the end of a trigram to the beginning of a trigram. This is done by the use of a node with the label of the Greek letter theta (Θ). This forms a cyclical relationship for the trigrams and simplifies the many-to-many matching process described later. By making trigrams into this special form of cycle individual trigrams are easier to identify.

In addition to all that has been said about these graphs, the variety introduced by the unification process as well as the other processes applied to the graphs allow them to be diversified. So instead of conducting matching on trigrams where every sentence has three nodes and therefore all sentences are three node trigrams, we have multiply connected trigrams structured in various ways. This introduction of variety in the structure of the graph allows for the fact that when trying to perform structural matches, there will not be match every time since every time there will not only be a three-node cyclical trigram, but a variety.

**The Use of Cycles for Cyclical Walks Over Graphs**

Sowa [11] “Speaking involves three stages: determining what to say, how to relate it to the listener, and how to map it to a string of words.”

In this context, the first part concerns selecting the parts of the graph to use, the second stage concerns selecting the starting and end points of the utterance path, and the third and final stage scans the graph and converts the structural data into words.

The second stage is generally a walk on the graph, walking over corresponding nodes and their edges and hopefully reaching the original node, completing a cycle over the graph. This is how the system in Sowa[11] is handled, and is the inspiration for creating a connecting edge allowing for the trigrams to be cyclical. The return to the original first node indicates that the trigram has been completed and the utterance path complete.

**Distinctions between parts of Speech**

There is a sharp distinction between articles, conjunctions, and auxiliaries that specify relations between objects and events (so called closed class elements) and open class words including nouns, verbs, and adjectives that make reference to specific objects and events. (S. Gress [6], pg 7) This distinction should be inherent in the graph structure.

**Graph Structure Review**

The graph structure is generally as follows. As above, each node is representative of a word. Also, however, we are considering including with each word a sub-graph associated with it constructed of its individual syllables for use in syllable matching and matching processes such as Grimm’s and Verner’s laws.

Word nodes are connected together in a trigram structure, with subjects, verbs, and direct objects forming the trigram and connections from nouns to adjectives where appropriate. Multiple trigrams, along with their frequency edges, construct entire sentence patterns. Sentences themselves that are mined from the dictionary are kept separately in a set with data mapping them to the syntactic graph containing the trigrams and syllables.

ORDERING OF UTTERANCE PATHS IN GRAPH

In various langauges, the “utterance paths” or ordering of the words in a sentence is important. For example, according to Sowa[11], there are three general types of ordering. Preorder languages such as Biblical Hebrew that put the verb first and nouns before the adjectives. Postorder languages such as Japanese which puts the verb last, nouns before adjectives, and postpositions after the nouns. Finallly Sow[11] describes Endorder languages. English and French, which put verbs in the middle, are considered Endorder languages. English however, has a postorder tendency to put the nouns after the adjectives. French is a closer approximation to an Endorder language, since it puts some adjectives before nouns and some after.

HANDLING OF RELATION VERBS OPPOSED TO CONCEPT VERBS

According to Sowa [11], “In English sentences generated from conceptual graphs, the verbs *be* and *have* usually correspond to relations rather than concept nodes.” In addition, languages like Russian does not require a verb and permits forms like ‘Bottles New’ or ‘At blithe babies fat bellies’. English also uses the word ‘do’ as a place holder. “As these examples illustrate, the same graph can be expressed in many different sentences, depending on the starting point and direction of the utterance path.” (Sowa [11])

USE OF A CONTEXT-FREE-GRAMMAR STYLE FOR ENCODING AND DECODING

Because of the variety of language structure, I suggest an encoding and decoding process similar to that used in APSG for processing known languages (Sowa [11]).

(sentence in language A is composed of noun phrases followed by verb phrases)

A:= {S}\*

S := NP VP (English, or another Endorder Language)

From these simple rules languages can be encoded into the universal, graph structure **R**, and decoded from **U** to meaningful text. ***Although the decoding would have to be determined through a graph matching process just as the nouns, verbs, and miscellaneous nodes and edges would have to be.***

For example:

E:= {S}\*

S := AP NP VP (English, or another Endorder Language)

(encoding a Japanese phrase according to the Preorder Language Rules Above)

J:= {S}\*

S:= NP AP VP

In actual APSG (Sowa [11]), these production rules are used to process grammar and include meta data as well such as data concerning whether the voice is active, the number of a noun, and the tense of the verb phrase. This information is not included in our graph because we are more interested in the syntactic connection between words than the semantic information that may be extracted from the paths in the graph (utterance paths).

However, here is an example of APSG taken from Sowa [11]

S(type (X) is AGNT) :=

NP (move AGNT := X, mark AGNT := X traversed;

case := NOMINATIVE  
 person:= person of referent (X)

number := number of reference (X)

VP (move AGNT:= X, voice:= ACTIVE)

tense := tense of S; mode := mode of S;

person:= person of NP; number := number of NP

A system such as APSG with its additional meta data would prove fruitful for incorporation into the existing Speech\*Morph system in future work.

Similar to the production rules in APSG above, Kamel [12] uses production rules embedded within a deeper tree structure to represent a sentence. An example graph is:

Sentence

Noun Phrase (NP)

Verb Phrase (VP)

Verb

Verb Object

Verb\_aux

Verb\_inf

Noun Phrase (NP)

You

Can

Erase

Non-Hidden

Files

Adjective

Noun

HANDLING DIRECT ARTICLES IN ENGLISH

The words “the” and “a” in English have many different uses, and according to Sowa [11] are difficult to generate. “At the first mention of an object the indefinite article “a” introduces it, but subsequent reference uses use the definite article “the” to refer back to it. Often, however, *the* is used to refer to an object that is implicitly introduced into the context:

*Do you have 1982 penny? I want to check the weight.*

(Here weight referring to the penny)

In other cases, both “the” and “a” may be used in both sentences:

*The horse is a noble animal.*

*A dog is an animal.*

MORE ON STRUCTURE

Sowa [11] “Heidorn created an NLPQ System using APSG rules similar to above. With it he was able to create very fluent, natural seeming blocks of text. (Emitted for text-length limitations). The same graph structure that was mapped into this English passage could also be mapped into other languages- computer languages as well as natural language. For NLPQ Heidorn wrote two sets of encoding rules. One set produced the English paragraph (emitted), and the other the set mapped the graphs into a program in the GPSS simulation language.

# Reduction and Query Reformulation

**Query Reformulation**

Query reformulation as mentioned in Ziqi Wang [4] et. al and their article “A probabilistic approach…” may be useful to condense the language graph upon or during construction. Query reformulation involves rewriting the original query with similar queries. Therefore phrases such as “NY Times” could be rewritten as “New York Times”.

A possibility would be an operator node as used in Zqi Wang et. al. pg 3 [4]. Upon reception of New York Times data or NY Times data, would map both signals to a single node, for example NY Times and New York Times to just New York Times. Multiple signals could be sent to this “operator” node to help condense the graph, remove obfuscation, and speed processing.

.

**Reduction of Neural Synapse Connections in the Human Mind**

The construction of neural synaptic connections within human beings during certain stages of language follows an interesting pattern, where the brain begins with a large number (in the trillions) of synaptic connections of primitive sounds. These sounds eventually become connected, forming words. Afterwards, the number of synaptic connections decreases significantly (500 million connections). This could indicate an isolative, condensing process where the synaptic connections are no longer representing individual sounds but entire words. (see Shara’s paper, [6] page 3). This process should be mimicked in the construction of the language graph per language for Speak\*Morph.

The growth of the graph should mimic statistical models of the brain as it learns a new language, steadily increasing from one word a week to one or more words a day. (Shara’s paper, [6] page 3)

# Graph Mapping (Matching)

Let a **complete language** be a language which when represented by a graph, attempts to contain at least one node for every noun, verb, and adjective available for processing.

The graph mapping between two graphs, for the purposes of this research, originates as being structurally dependent and language independent. Once the source Rosetta graph **R** is constructed via dictionary methods, the identification of words and possibly sentences and syllables should be conducted through a graph matching of the individual graphs **R** Rosetta and **U** incomplete (unknown). That is to say, during the matching process similar nouns, verbs, and adjectives may be used to assist in the matching process after the structural matching processes have been completed, or as heuristics to aid in the graph matching process. After the matching process is complete, the matching **M** may be used to determine according to its results to translate from **R**’s semantic node over to **U**’s semantic node and thus performing a translation.

Given two graphs, A=(VA, EA) and B=(VB, EB), each representing a complete language let there exist a mapping between the two B = (A), such that () represents a graph matching algorithm for the two given language graphs. This mapping may use several subroutines for itself, such as usage of Verner’s Law, Grimm’s Law, graph completion using word context, frequency of association between words, properties inherent in the graph, and any other logical and useful subroutine.

Therefore we would also have….

Set of Sentences

CA

Set of Sentences

CB

Fig 1.1

That is, there is a mapping between Graph a and Graph b labeled as as there is a mapping between the set of sentences also, where is similar to in some way, possibly by some linear constant k

Further Illustration of mapping:

“Bob”

hungry

Car

Unknown

**Incomplete Graph U**

**Complete Graph R**

( MAPPING FROM ABOVE TO BELOW**)**

Fig 1.2

**Graph Matching in General and Applied to Speech\*Morph**

There are two approaches to mapping between the language graphs. Typically, if there wasn’t data specific to individual nodes or edges a graph-matching algorithm would be used that relies entirely on the structural information of the graph. There is a plentiful number of algorithms used as such.

However, the graph we are matching against, **U**, by default contains at least one node, but most likely many nodes and edges, that are already established with content as well as our graph **R** which contains complete dictionary content. This means that our graph matching algorithm from above that relies entirely on structural information can be augmented by the node, edge, and edge frequency information available for comparison of the graphs **R** and **U.**

The goal of the matching process being that once we know a direct mapping from graph **R** to graph **U**, the nodes of **U** whose semantics are unknown (for example, given an arbitrary word **laku** that we do not know the meanings of), now the mapping will have assigned a translation value to the word **laku**, which in this hypothetical case would mean the world “lake.”

**Graph Mapping using the Island Method**

As described later, the matching process between language graphs may begin at the syllable sub-graph level, then move on to words, trigrams, and finally sentences. An example process of this would be the **Island Method** referenced in Williams [18]. In such a mapping, the algorithm begins with random single nodes throughout the graph that is being matched, with the nodes growing into cliques and further more into larger and larger sub-graphs until a match is found. If the map, once extended beyond sub-graphs to sentences is matched successfully, the algorithm is considered successful and complete.

**Levels of Graph Matching**

The graph matching process in general progresses in this order, going from Syllable Matching first to Sentence matching in the worst case scenario:

*Syllable Matching*: Syllables between graphs are attempted to be matched. This generally follows natural language rules such as Grimm’s Law.

*Word Matching:* Words are attempted to be matched, node to node between graphs.

*Trigram Matching:* In this instance, trigram patterns are attempted to be matched. Similarly structured trigrams with one or two missing components are matched and their similarity measured.

*Sentence Matching:* In the case where a subgraph cannot be matched via syllables, words, or trigrams, entire sentences are attempted to be matched. This, as well as trigram matching, depends heavily on the structure of the related graphs.

**Grimm’s Law and Verner’s Law and Sylllable Differences**

There are many different syntactic differences between languages (see Grimm’s Law, Verner’s Law, Lee Osterhout [3] pg 513) that may be represented through mapping of trained graph data sets, each graph representing a language and the mapping the translation of one language to another, which may in some way correspond to the isomorphism of the two graphs

**Syllable Subgraphs**

It is suggested in [5] Huang, Wang, Lee, etc. al in “A Mandarin Speech Dictation System”, to use subgraphs to represent syllables of entire words for the Mandarin Dictation system built in the article. Also, a neural network with DP based warping capability and Bayesian decision theory; (DPBNN) is used. These two things may prove useful in the Speak\*Morph application. Dividing words into syllables as subgraphs could provide an important tool, considering the mapping heuristics for the graph matching is mostly based at the syllable level (see Verner’s and Grimm’s laws…)

In Huang, [5], “firstly, mapping of syllables to homonym characters is conducted. Then word formulation and category assignment are performed to remove invalid characters (again, this is a dictation system but the idea of removal of invalid data may play an important part in translation of incomplete languages). Finally, statistical analysis is propose and performed to select the best word sequence.”

Continued in [5] Huang, “A dynamic programming Bayesian neural network was proposed in this paper to recognize the basic primitives (vowel, consonant, and lexical tone).” Again, these ideas tend to be promising towards using such system elements for translation of incomplete langauges.

**The Graph Edit Distance**

Pawarand Zaveri in “Graph Based Pattern Matching” [1] pg. 1023 describe the graph edit distance. The edit distance of G and G’ is the shortest series of operations that transform G into G’. The shorter the series of operations is, the more similarity in graphs. In fact, graph isomorphism, subgraph isomorphism, and maximum common subgraph are all special means of graph edit distance computation with special cost functions.

Graph Edit Distance is obtained by transforming Graph 1 (G1) into Graph 2 (G2), applying a sequence of edit operations to G1 (node deletion/insertion/substitution and edge deletion/insertion/substitution). Each edit operation is assigned a cost and the cost of a sequence of operations is the sum of the individual edit costs. Distance is therefore determined by the minimum cost to transform G1 to G2. (Pawar and Zaveri)[1] (The sum of the cost of node insertions, deletions, substitutions and the sum of the cost of edge insertions, deletions, and substitutions.)

(Pawar and Zaveri) [1]

[Ulman]’s backtracking method for Graph Matching.

**Earth Mover’s Distance**

A more efficient and successful metric for determining many to many graph matching may be the use of Caterpillar Trees and Euclidean Graph Embedding (or Spherical Embedding) as proposed by Shokafundaeh and Dimicri along with the Earth Mover’s Distance to determine the actual matching between graphs. EMD has been shown to be more successful than Graph Edit Distance.

**Shared Syntactic Structure**

It has been shown that, “Bilinguals appear to have shared syntactic representations for similar constructions between languages but retain distinct representations for noncognate translation equivalents.” [9]

Multiple studies demonstrate that in bilinguals, multiple languages will share syntactic structure between languages for common nodes. This is significant for the problem of mapping, since the two graphs were are presented with, in theory, are not necessarily two distinct graphs but possibly two graphs that share nodes and edges. Z. Cai [9] goes on further to say that most bilingual speakers have a representation that is at least partly integrated across languages.

In [9], the graphs are not only integrated together (one or more languages) but are fully integerated as one graph, with origin nodes related to their respective language. For example,

Mandarin

Cantonese

Syllable

Shared Syllable

Syllable

[9] also suggests that individual language graphs could grow from one into another, suggesting “A child learning Cantonese and Mandarin simultaneously would usually find that a word in one language has an equivalent in the other language that has minor phonological differences and could be used similarly.”

[9] continues to state that, “bilinguals collectively represent similar syntactic constructions across languages so that using a construction in one language facilitates the use of a corresponding construction in another language via the same syntactic representation.” One can deduce from this idea that multiple languages whose nature is structurally similar may be represented by a similar data structure.

I propose that a **language space** exists, that is, a space consisting of individual graphs, each representing a language and mappings between languages.

**Verb (Multiple Language) Connections (Also shared Syntactic Structure)**

In Z. Cai [9] it is evident that, since there is a stronger priming boost for sentences across languages with the same verb, that the possibility exists that languages in a language space should share verbs, as corresponding verbs seem to have a stronger connection factor than nouns, articles, and other parts of speech.

F. Bouzit [10] uses verbs as the central portion of his machine translation system and the Fillmore theory. “The idea is to use the verb as the kernel of the sentence and to study the role of its other constituents (nouns) with this kernel.” This is because of the fact that no language is ordered precisely as the other. “If two sentences have different representations, they may transport the same meaning.” [10].

F. Bouzit [10] also uses different meta-data attached to the parts of speech, such as Agent, Object, or Instrument for nouns, Action or Purpose for verbs, etc. There is more discussed later concerning F. Bouzit’s frame based method.

# Combination of language graphs to generate Rosetta Graph R

Let us denote the graph that will be used for universal translation as the Rosetta Graph, R.

It may prove to be useful to construct the graph R from a sequence of individual graphs unique to a variety of already established languages (such as Russian, Cantonese, English, German, and French…)

Use of individual graphs, accepting training data to tailor their graph to a specific language and the using the relationships between graphs to generate the Rosetta graph R (possibly generating **R** using training data from all input complete languages) and following completion of the isomorphic graph S, where S is a graph of the incomplete language in the process of translation.

(Ziqi Wang, et. al page 7 right column) [4] describe word pair mining and mapping from one query (word) to another query (word). This inspires a similar process as mentioned above, where individual language graphs could be created via training data (Ziqi Wang et. al. page 4) [4] and a transition function be created from language graph to language graph representing a mapping from graph to graph. A combination of the mappings and graph content from each graph could be summed over to form a general Rosetta graph R.

# Machine Learning: Training Rosetta Graph R via Dictionary Data

**Using Generic Symbols in Training Data**

As is performed in Gavrilov [13], it may be fruitful to instead of pushing rules into knowledge base, to train the knowledge base using typical machine learning procedures as in Neural Networks and Dynamic Bayes Nets. In doing so, Gavrilov recommends using certain symbols such as a   
“-“ or the “@” sign to isolate certain words in the training process that should not be included in the training data (such as the word “the” or the word “a” which would appear multiple times and could overload the network).

# Architecture of Application

**Comparison between Laskri’s Translation Frame Approach and the use of subgraphs**

Laskri et. al. in “Multilingual machine translation…” [10] on page 157 uses the frame based representation of Minsky to translate from Arabic to another language. The process I am presenting is similar, however, portions of the graph (subgraphs or paths) are used to represent the frames between languages, and the translation portion is handled by mapping between subgraphs.

An example illustration is below:

Sentence in Arabic (path)

Sentence in Target Language

Construction

Analysis

Frame in target language (Subgraph)

Translation

(Mapping)

Frame in Arabic (Subgraph)

**Application execution flow**

1. Create a data structure of hashmaps termed “Word\*Map”.
2. Read in from dictionary for selected language (English for example). Generate graph from dictionary input (Multi-Lexicon database)
   1. Construct nodes in **R** and **U**, one vertex for every word pulled from Multi-Lexicon, one vertex per other associated vertex in graph **U.** Store associations from Multi-Lexicon in Word\*Map hashmaps.
   2. Parse sentences from a sentence source **S** (such as a written novel, random web crawl, etc.), (training) increasing frequencies between edges of trigrams when given two words connected by an edge in the graph are identical to two words in the dictionary input. (Wikipedia link sent to me by Prof. Sultanik).
   3. Perform unification process as necessary.
3. Repeat (1) until satisfiable **R** is generated. *Possibly train a second graph(s)* ***Rn*** *and link* ***Ri’s into aggregate R,*** *where each* ***Ri*** *represents an individually trained graph of a language, and the aggregate* ***R*** *represents the graph over all trained languages.*
4. *Perform graph Reduction and Query Reformulation as described in the above text to increase efficiency [cite]*
5. *(Conjecture) A method to increase the accuracy of the graph mapping would be to perform the following graph matching not only on the trigram level but starting on the syllable level, moving to word level, then trigram, and to sentence and so on... [cite] Use of Grimm’s Law, Verner’s Law, etc. in order to match word structure transition. Possibly for future work.*
6. Perform graph matching procedure
   1. (Match structural content) Perform structural graph matching as described in Dr. Ali’s papers.
   2. (Match node content) For each structural map, send structural graph matching process through verification filter.
      1. Verification filter processes each unknown node, certifying that the known nodes are matched semantically by cross referencing the known word’s nodes in each graph using the Word\*Map.
   3. (Match frequency content) Process Frequency Filter. Frequency filter matches the frequencies between word nodes in languages. If frequencies are close (within a certain given constant percentage), and the above two filters are approved, process moves to the next phase.
7. Augmenting Phase. Parts a,b, and c of (5) generate a score. Unknown nodes, according to the graph structure and ordering from above, are assigned values from known nodes in the Rosetta graph **R** if the score generated in (5) is sufficient. Missing edges between known nodes are applied similarly according to the graph structure and ordering above. Process jumps to (5) and continue matching/filtering/augmenting process until there are no more known structural mappings between **U** and **R**, a.k.a. no more known graph matches.

For empirical values (statistics)

(Keep Training Set and Test Set separate)

Training Set (several books in partial Italian and English)

Test Set (a partial book in a known language that is not Italian)

First training tests English against Italian to determine a relative success rate.

Second training tests English against Sumerian to see what sort of results we get.

**ALSO**

**There are Sumerian Lexicons and Dictionaries online**

**(Search: Sumerian Words). Do a trial run after the above empirical data against a Sumerian Word Database and report results.**

So download **Sumerian Lexicon (word list), Sumerian Dictionary (for hashmapping (less words than lexicon), and a myth in native Sumerian (for “book” training data”)**

**Necessary Data Structures**

**Class WordMap**

**{**

**hashmap<string< vector<integer> > R\_nodes;**

**hashmap<string< vector<integer> > U\_nodes;**

**}**

**Example use of WordMap:**

Love

Word\*Map (HashMap)

[ 1, 5, 8, 12, 13, 19]

Lukol

Word\*Map (HashMap)

[ 0, 2, 8, 7 1, 19]

1. If “Love” in English is in the source node of **R**, pass it as key into the hashmap R\_nodes to return a list of vertices with the word love as its semantic tag (that is, those vertices are labeled with the word “Love”.
2. If node labeled “Lukol” is a possible match in the semantic filtering process, pass “Lukol” to the hashmap U\_nodes. It will return a list of vertices, as seen above. If any of the vertices match than Love and Lukol are probable synonyms across languages. If the current vertex resides in the vertex list retrieved from the two hashmaps, then the word Love is a match to the word Lukol and the algorithm returns success.

**Class Graph R (standard graph data structures)**

**{**

**vector<Edges> edges;**

**vector<Nodes> nodes;**

**}**

**Class Edges E (standard edge data structure)**

**{**

**double frequency;**

**Node \*incoming\_node;**

**Node \*outgoing\_node;**

**}**

**Class Nodes V (standard vertex data structure)**

**{**

**vector<Edges\*> outgoing\_edges;**

**vector<Edges\*> incoming edges;**

**vector<Edges\*> undirected edges;**

**Quantifier q;**

**string label\_data; (can be a logical operator or theta as well)**

**}**

**Class Quantifier q (quantifiers to assign to vertices)**

**{**

**enum quantifier\_type = { universal, existential, verb}**

**quantifier\_type quantifier;**

**}**

**Necessary Global Operations (functions)**

* **Create graph R() (complete graph)**
  + **Read words from dictionary file and create nodes, creating graph R**
  + **Use queries from dictionary file into MultiWordNet to find correspondences between words in languages and store in Word\*Map data structure**
  + **Parse input text document (from Dr. Sultanik’s link) and add edges to nodes**
    - **Add existential and action quantifiers**
    - **Unify existential quantifiers and action quantifiers**
  + **Continue to train graph frequencies from text document**

**until training is satisfactory.**

* **Create graph U() (incomplete graph)**
  + **Perform all below to a certain threshold to create an incomplete graph for testing:**
  + **Read some words from dictionary and create nodes**
  + **Parse input text document (from Dr. Sultanik’s link) and add edges to nodes**
    - **Add existential and action quantifiers**
    - **Unify existential quantifiers and action quantifiers**
  + **Continue to train graph frequencies from text document**

**until training is satisfactory.**

* **Perform unification on both graphs R and U over universal quantifier.**
* **Begin Graph Matching Process**
  + **Match structural data in graph (Structural Filter)**
  + **For each structural mapping,**
    - **Process semantic filtering process – that is, matching of like nodes using Word\*Map to determine if sub-graphs determined by structural mapping are indeed matches semantically.**
    - **Perform process frequency filtering process – that is, matching of like edges. If frequencies on connected edges of Structural sub-graph are close enough (within a certain of threshold) .**
    - **If all three filters are passed successfully, assign unknown word node in Word\*Map to the matching meaning listed in the Word\*Map. *Unknown word matches known word.***

**Graph Creation Procedure**

There are three major steps in the creation of the graph.

First, allocate a node for every word in the dictionary file that Dr. Sultanik gave me the link to.

Second, query the sql database for MultiWordNet to find correspondences between words in different languages, and store those correspondences in the Word\*Map hashmaps structure.

Third, train the graph using arbitrary text documents. The Wikipedia project one that Dr. Sultanik supplied in his email should work.

**Structural Mapping Procedure (according to Dr. Ali and Yusuf)**

Graph is first embedded into a tree. Then the tree is decomposed using caterpillar decomposition. Using the paths resulted from caterpillar decomposition, he embeds vertices into vector space using spherical embedding method. Once we get the vertices embedded into euclidean space, now we can apply earth movers distance algorithm to determine the cost of closing the holes with the piles. The little the cost is, the higher the matching rate is.

# References

**Bibliography**

**Read Graph/Language Articles**

[1] V. Pawar and M. Zaveri, “Graph Based Pattern Matching”, 2011 Eight International Conference on Fuzzy Systems and Knowledge Discovery, pp1022-1026

[2] F. Zhou and X. Yu, “Graph-based Language Model of Long-distance Dependency”, 2011 International Conference on Asian Language Processing, pp37-40

[3] L. Osterhout, et. al., “Second-language learning and changes in the brain”, Journal of Neurolinguistics. Vol. 21, (2008) 509-521

[4] Z. Wang, G. Xu, et. al., “A Probabilistic Approach to String Transformation.”, IEEE Transactions on Knowledge and Data Engineering. pp2-14

[5] C. Huang, J-F Wang, J-Y Lee, “A Mandarin Speech Dictation System based on Neural Network and Language Processing Model” Chang Gun College of Medicine and Technology, Taiwan. pp 437-445

[6] S. Gress, “An Analysis to Contributing Factors of Language Acquisition Ability”, Pennsylvania State University.

[7] “Verner’s Law”, Wikipedia

[8] “Grimm’s Law”, Wikipedia

[9] Z. Cai, M. Pickering, H. Yan, H. Branigan, “Lexical and syntactic representations in closely related languages: Evidence from Cantonese-Mandarin Bilinguals.” Journal of Memory and Language, vol. 65 (2011) pg 431-445

[10] F. Bouzit, M.T. Laskri, “Multi-Lingual Machine Translation System Based on a Linguistic Approach.” 2011 Fourth International Symposium on Innovation in Information & Communication Technology

[11] John Sowa, “Generating Language From Conceptual Graphs”, IBM Systems Research Institute, pp 29-43.

[12] M. Kamel, Y Quintana, “A Graph Based Knowledge Retrieval System”, Pattern Analysis and Machine Intelligence Group at University of Waterloo. pp 270-274.

[13] A. Gavrilov, “A Combination of Neural and Semantic Networks in Natural Language Processing”, Proceedings of the 7th Korea-Russia International Symposium, KORUS 2003

[16] L. Chen, A. Gupta, M. Kurul, “Efficient Algorithms for Pattern Matching on Directed Acyclic Graphs”, Proceedings of the 21st International Conference on Data Engineering, ICDE 2005

**Unread General Graph/Language Articles (Add to Related Work?)**

[14] Marco Gori, “Exact and Approximate Graph Matching Using Random Walks”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 7, pp. 1100-1109

[15] R. Navigli and M. Lapata, “An Experimental Study of Graph Connectivity for Unsupervised Word Sennqnnse Disambiguation”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no 4, pp 678-691

[17] J. Cheng, J. Xu Yu, B. Ding, et. al, “Fast Graph Pattern Matching”, ICDE 2008, pp 913-919

M. Meurs, F. Lefevre, and Renato de Mori, “Spoken Language Interpretation: One the Use of Dynamic Bayesian Networks for Semantic Composition”, ICASSP 2009, pp 4773-45776

C. Mellish, J. Pan, “Natural language directed inference from ontologies”, Department of Computing Science, University of Aberdeen. 2008.

A. Bishop, B. Ristic, “Fusion of Natural Language Propositions: Bayesian Random Set Framework”, Australian National University, 2011.

J. Taylor, V. Raskin, L. Stuart, “Matching Human Understanding” 2012 IEEE International Conference on Systems, Man, and Cybernetics. October 14-17, 2012. pp1130-1135.

I. Grunsky, I. Potapov, et. al, “On algebera of languages representable by vertex-labeled groups”, Institute of Applied Mathematics and Mechanics, 2010. pp 43-48

E. Khorasani, S. Rahimi, et. al. “Formalization of Generalized Constraint Langauge: A Crucial Prelude to Computing with Words,” IEEE Transactions on cybernetics, vol 43. No1. February 2013.

**Read but unreferenced (References Given by Yusuf)**

*Professor Ali’s Papers*

M. Fatih Demirci, Yusuf Osmanlioglu, Ali Shokoufandeh, Sven Dickinson. “Efficient many-to-many feature matching under the l1 norm.” Computer Vision and Image Understanding (2011)

M. Fatih Demirci, Yusuf Keselman, Lars Bretznerm Sven Dickinson. “Object Recognition as Many-to-Many Feature Matching.” International Journal of Computer Vision v69(2) 203-222 (2006)

*References linked to Professor Ali’s Papers*

Jiefeng Cheng, Jeffrey Xu Yu, et. al. “Fast Graph Pattern Matching” The Chinese University of Hong Kong, China. ICDE 2008.

Marcello Pelillo, Kaleem Siddiqi, Steven W. Zuckewr. “Matching Hierarchical Structures Using Association Graphs”. IEEE Transactions on Pattern Analysis and Machine Intelligence Vol. 21, No 11. Nov. 1999

Rucga Agarwala, Vineet Bafna, et. al “ON THE APPROXIMABILITY OF NUMERICAL TAXONOMY (FITTING DISTANCE BY TREE METRICS” 1999 Society for Industrial and Applied Mathematics. Vol. 28 No. 3 pp 1073-1085

[18] Mark L. Williams, Richard C. Wilson, Edwin R. Hancock “Deterministic search for relational graph matching”. Pattern Recognition. V 32. (1999) 1255-1271