Tenth Annual
Bay Area Discrete Mathematics Day

PHIL
The Probabilistic Hierarchical Inferential Learner

April 9th, 2005

Ruchira S. Datta
Google, Inc.

Acknowledgements

PHIL was invented by
Georges Harik and Noam Shazeer
of Google, Inc.

US Patent Application 0040068697

Search for [Georges Harik Noam Shazeer]

Includes everything in this talk, and much more!

Understanding The Meaning of Text

A text is a bag of words, such as a webpage.

The person who created the text had certain concepts in mind.

To understand the meaning of the text,
we want to model the concepts.

Model them by their effects:
when one thinks of a concept, one

- writes some words
- thinks of other concepts

Phil Network

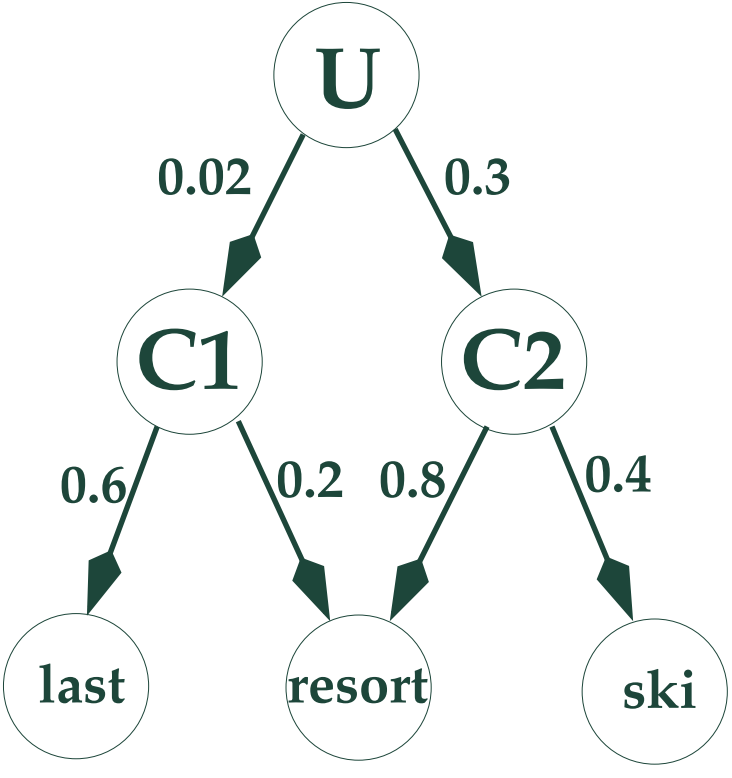
A Phil network is a directed acyclic graph,
oriented from top to bottom, with weighted edges.

At the top is a single node, the universal concept.

At the bottom are the leaves or terminals,
which represent either words,
or non-compositional compounds
such as "new york" or "san francisco".

In between are other nodes representing concepts or clusters.

Example of a Phil Network



A Generative Model of Text

We can execute a Phil network to generate some text.

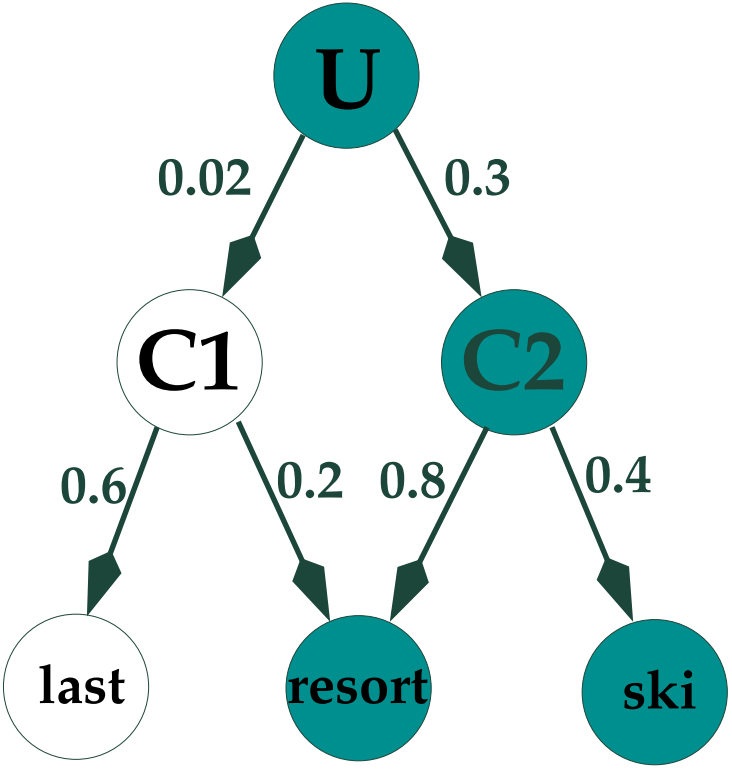
We always begin by firing or activating the universal node.

We throw dice to determine whether to fire other nodes.

The chance that a node will fire depends on
the activation of its parent nodes and
the weights of the links from its activated parents.

When a terminal fires,
the corresponding word or compound occurs in the text.

How To Generate A Ski Resort



What Can We Do With Phil?

We can compare the concepts occurring in queries, documents, and ads.

We can compare the concepts between two documents, and
form a distance metric for clustering similar documents.

For a query with ambiguous meaning,
we can present results corresponding to the alternate meanings,
in proportion to their likelihood.

We can use the concepts as features in order to classify texts.

We can guess whether words are misspellings of each other
based on the concepts they induce.

The Problem of Document Length

Suppose we had a concept for countries of the European Union.

Suppose when this concept is active,
it activates any country with probabililty 1/25.

Then texts with more countries become
more and more improbable.

A list of all the countries becomes almost impossible.

The Solution

During execution, we pick an activation level for each cluster.

The probability that a link of weight W,
leading from a concept C with activation level $A \geq 1$
to a terminal T,
will cause T to fire is:

$$1 - \frac{1}{e^{AW}}$$

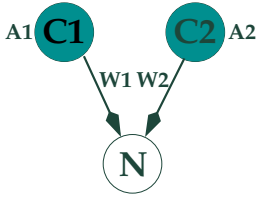
Increasing the activation level lets us generate longer texts.

Noisy-Or Bayesian Networks

Nodes in a Phil network are fired independently by each parent.

The probability that a node is not fired is
the product of the probabilities that each parent does not fire it.

The node uses "noisy-or" to combine the link activations from each parent.



The probability that N is fired is $1 - e^{-(A_1 W_1 + A_2 W_2)}$.

A Phil network is a Bayesian network with noisy-or combinations.

Inference Using Loopy Belief Propagation

Given a text (a set of terminals), how do we infer the concepts?

Send down messages from parent to child,
stating our belief in parent due to everything but messages from child.

Then child updates belief in itself, according to link activation from parent.

Send up messages from child to parent,
stating the belief in the child due to everything but messages from the parent.

Then the parent updates its prior belief in itself, according to Bayes's rule:

$$P(A|B) = P(B|A)P(A)$$

Learning The Phil Model

A single text can be explained by a single concept
which links exactly to the terminals occurring in the text.

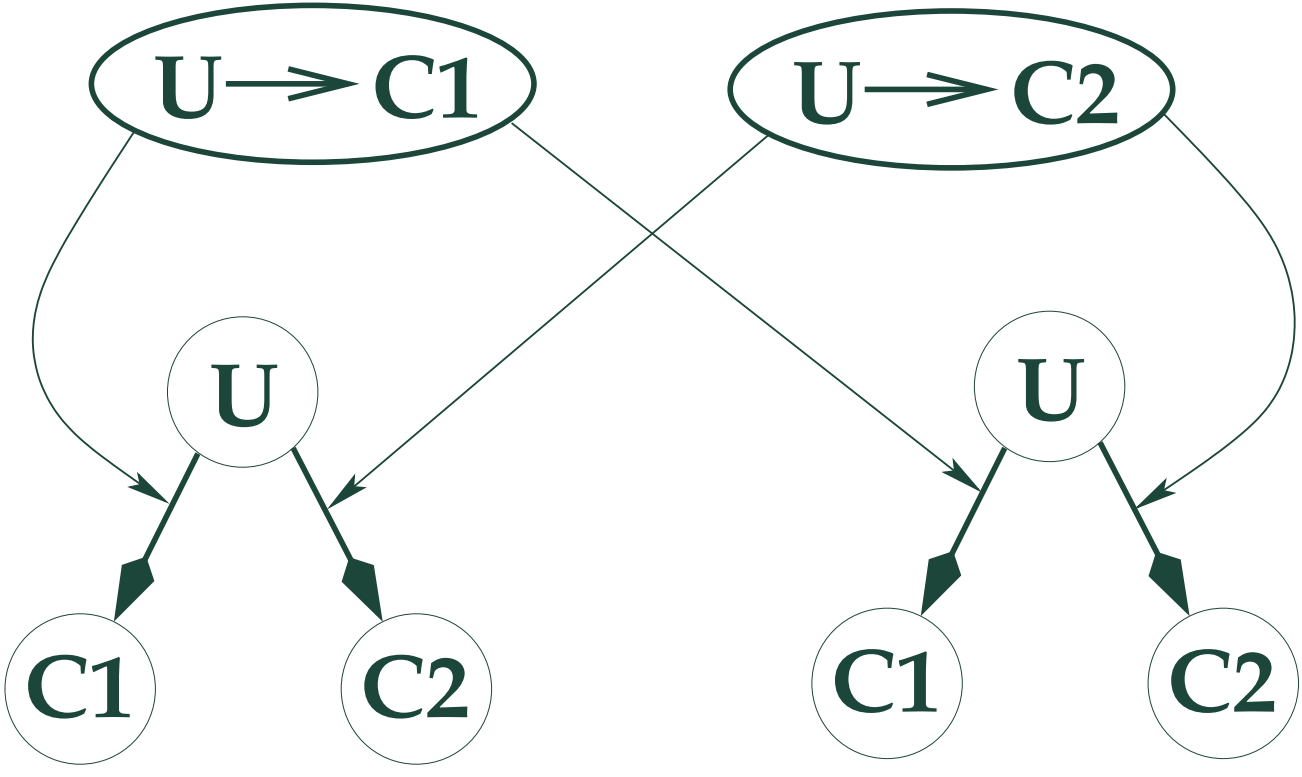
We learn the Phil model from a large number of training texts.
For each text, we replicate the whole Phil model into a local network.

Then we introduce global nodes.
A single global node points to corresponding links in each of the local networks.

We run inference using loopy belief propagation on the big network,
to determine whether each link should exist.

We solve a one-dimensional optimization problem to find the best link weight.

Example of an augmented Phil network



Non-compositional compounds

Look at how often a sequence of words occurs together
versus how often its subsequences occur separately

Build a lexicon of non-compositional compounds:

Static compounding: Use dynamic programming to go through the text.
Find the decomposition which maximizes the probability
of the sequence of words seen so far.
Then check whether the best decomposition including the next word
uses that word separately, or to end a non-compositional compound
including some of the immediately preceding words.

Dynamic Compounding

And Now For The Demo...