# Тощев А.С. Интеллектуальная система повышения эффективности IT службы предприятия

## Формальная модель

Формальная модель системы представляет собой слабосвязанные вероятностные Машины состояний (или конечные автоматы), действующих параллельно и управляющихся также вероятностной машиной состояний.

Например, GoalManager, компонент, который, устанавливает цели в системе, согласно текущему контексту. Управление машинами идет по событийной модели. Например, событие «пришел инцидент от пользователя», параллельно активируются машины, отвечающие за классификацию (ClassifyIncident critics). Каждая машина работает параллельно и возвращает результатом своей работы вероятность, что активирует новое событие в системе, которое обрабатывает основная машина (далее TLC).

Далее рассматривается работы данной машины на примере.

**Входные данные:**

Запрос: *Please install Firefox.*

Начальное состояние: *все концепции известны*.

1. Приходит инцидент от пользователя;
2. TLC выставляет цель ProcessIncident. К каждой цели привязан набор критиков (каждый критик действует как отдельная вероятностная машина);
3. Запускается машина PreliminarySplitter, затем в RelationExtractorKB, где подвергается обработки LinkGrammar NLP, который преобразует данное предложение в граф, результат лексического и синтаксического разбора, на выходе получается следующий граф:

\_advmod(install, please)

\_obj(install, Firefox)

pos(install, verb)

inflection-TAG(install, .v)

tense(install, imperative)

pos(please, adv)

inflection-TAG(please, .e)

pos(., punctuation)

noun\_number(Firefox, singular)

definite-FLAG(Firefox, T)

pos

pos(Firefox, noun)

pos

obj

Firefox

Install

pos

advmod

Please

*Рисунок 1. Граф, представляющий семантико-лексический разбор предложения*

*Модуль содержит ограничение на количество предложений и количество слов. Длина предложения не более 1024 символов.*

1. Граф, изображенный на Рисунке 1 подвергается разбору и преобразуется во фразы, которые группируются в предложения. В исходном примере нет идиом, но если такие встречаются, то они также будут составлять одну фразу. Отношения, начинающиеся с подчеркивания – это связи между нодами графа, остальное это описания конкретного слова. Ссылка может идти также не на лист, а на куст дерева, для этого алгоритм построен рекурсивно. У каждого нода графа есть именнованые ссылки, в нашем случае мы берем только ссылки вида: "\_subj" (подлежащие); "\_obj" (дополнение); "\_iobj" (косвенное дополнение); "\_advmod" (наречие); "of" (принадлежность). На выходе мы получаем граф, перефразированный в рамках объектов приложения - AnnotatedPhrase. Кроме того, будут отфильтрованы лишнее связи, также копируются свойства pos (часть речи), tense (время);

Firefox (noun)

Install (verb, imperative)

Please (Ignored)

Рисунок 2. Предложение в виде графа после обработки.

1. Результат передается в LinkParser, который ищет соответствие между словами, выявленными в разборе и базой знаний; Например, install уже есть в базе и он получит ссылку на концепцию в базе. Каждая концепция в базе поддерживает свойства generalization, specialization. Рассмотрим концепцию install;

Generalisation

Action

Install

Рисунок 3. Пример абстрагирования концепта.

1. Выставляется цель ClassifyIncident и запускаются машины классификации. Например, машина DirectInstrutctionAnalyser. Первым этапом он смотрит есть ли подлежащие в системе подлежащие. Во фразе их нет, значит итоговая вероятность не уменьшается, иначе взвешенный результат будет меньше, так как данный критик нацелен только на действие и объект, над которым нужно его совершить. Далее ищется действие в количестве 1 и прилинкованная к ней концепция объекта. Критик содержит в себе набор правил, который обрабатывается логической машиной PLN. На вход подается граф, изображенный на Рисунке 2. Например, правила вида «Граф содержит концепцию подлежащего» или «У нода графа действия, есть связь с нодом типа дополнение». В интерпретаторе правил есть поддержка прямой логики ”forward chaining”: конъюнкция, дизъюнкция, отрицания, равно, меньше.;
2. Так как решения еще нет в базе, то начинается поиск решения, путем сравнения графа исходной проблемы и хранящегося в базе знаний решений. Во время сравнения изоморфизма графов исходной проблемы. В результате подсчитывается коэффициент d – 0 графы идентичны. Во время подсчета учитывается ссылка на обобщенные концепции. Например, если есть две концепции Winrar и Archive, обе ссылаются на базовую концепцию Software, то соответствие данной вершины будет 0.5, в зависимости от отдаление базовой концепции соответствие будет падать: 0,75; 0,865 и т.д.;

Install (verb, imperative)

Winrar (noun)

7zip (noun)

Install (verb, imperative)

Archiver

Archiver

Рисунок 4. Сравнение графа входной проблемы и имеющегося в базе знаний описания проблемы.

В данном случае d=0,5.

Решения буду ранжироваться по увеличению d.

**Входные данные:**

Запрос: *The installation of Winrar that I got this afternoon did go wrong. During installation nothing else was running. When I tried to start Winrar I got the fault message that is attached here.*

Начальное состояние: *все концепции неизвестны*.

Разбор данного запроса будет представлять следующий граф:

Installation

Winrar

Go wrong

pos

verb

pos

Рисунок 5. Разбор второго запроса.

**Входные данные:**

Запрос: *I have Office 2010 installed, but I need office 2016.*

Начальное состояние: *все концепции неизвестны*.

Разбор данного запроса будет представлять следующий граф:

I

need

have

pos

verb

pos

Office 2016

Office 2010

Рисунок 6. Разбор запроса с желаемым состоянием.

В данном случае сработает Критик для проблемы с желаемым состоянием. Он запустит путь мышления для выявления текущего состояние и желаемого состояния. Далее будет также произведен поиск решения сначала для удаления текущего состояния, потом для установки нового состояния.