ВЕБ-СИСТЕМА ДЛЯ АДАПТИВНОГО ПОИСКА СОТРУДНИКОВ IT-КОМПАНИЙ НА ОСНОВЕ ИХ КОМПЕТЕНЦИЙ ПО ЛОГИЧЕСКИМ ВЫРАЖЕНИЯМ

Савенко А. Г., Шерстнев А. С.

Институт информационных технологий Белорусского государственного университета информатики и радиоэлектроники, Минск, Беларусь,
e-mail: savenko@bsuir.by

В настоящее время в бизнес-процессах предприятий все более важную роль стали занимать вопросы управления персоналом и подбора новых сотрудников. Особенно актуален этот вопрос в сфере информационно-коммуникационных технологий, так как данная задача может быть усложнена еще тем, что иногда не всегда понятно, какими компетенциями должен обладать сотрудник на момент начала работы над проектом.

Для решения проблемы подбора персонала для работы над проектами IT-компаний были разработаны подход подготовки, обработки и загрузки данных о компетенциях, а также гибкий алгоритм, который позволяет осуществлять поиск по загруженным данным. Гибкость алгоритма поиска достигается путем предоставления пользователям возможности формировать логические выражения из компетенций и автоматически формирует нужные обобщения компетенций в один класс, по которому в дальнейшем можно будет осуществлять поиск.

С точки зрения информационной модели системы реализуется сложный поиск на основе логических выражений. В данном случае целесообразно хранить информацию в виде связей между сущностями, при комбинировании которых можно добиться нужного критерия поиска. Сами сущности должны описывать как можно меньший объем информации для того, чтобы обеспечивать меньшую гранулярность и, следовательно, предоставлять более точные результаты. Связи должны быть односторонними и направленными, а их количество должно быть минимальным. Пример модели графовой базы дынных в разрезе одного сотрудника представлен на рисунке 1.

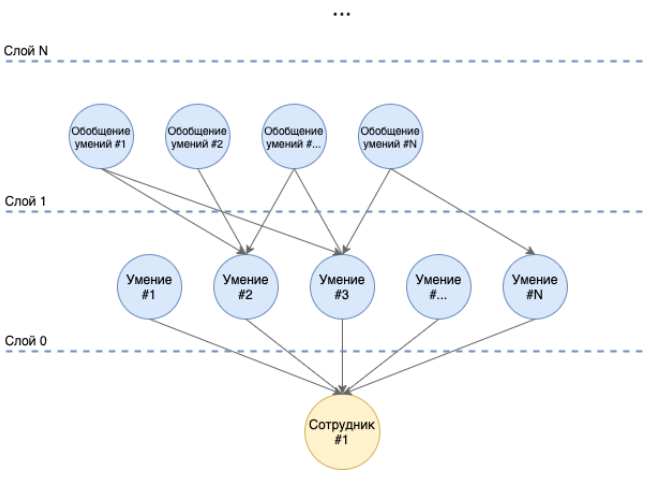

Рис. 1. Модель графовой базы данных

Как можно видеть из базовой модели, данные организованны в виде слоев. На самом нижем уровне находится узел с информацией о сотруднике. Данный узел связан с узлами умений односторонней связью. В свою очередь узлы умений связанны со следующим слоем обобщений данных умений в более широкие группы понятий. Данное обобщение позволит системе поиска находить сотрудников по более широким понятиям их компетенций. Так же на одном условном слое связи между узлами запрещены, чтобы предотвратить цикличность при поиске.

Важной задачей при разработке адаптивного алгоритма поиска по компетенции является быстродействие – необходимо обрабатывать большой объем данных за приемлемое время ожидания пользователя, а также гибкость в построении запросов поиска. Исходя из этого, необходимо рассмотреть пограничные случаи работы приложения, при котором поиск будет максимально неэффективным.

Количество связей между узлами двух слоев на заданном уровне определяется функцией $L(k, m, n)$, где k – количество узлов первого слоя, m – количество узлов второго слоя, n – номер уровня и при этом $k > 0$; $m > 0$; $n \geq 0$; $k, m, n \in N$. Количество узлов следующего слоя задается функцией $G(t, n)$, где t – количество узлов предыдущего слоя, n – номер уровня и при этом $t > 0$; $n \geq 0$; $t, n \in N$. Общее количество связей в многослойном графе, с ограничениями на связи описывается формулой 1.

$$R(t,n,L,G) = \begin{cases} R(G(t,n), n-1, L, G) + L(t, G(t,n), n), & n > 0 \\ 0, & n = 0 \end{cases}, \quad (1)$$

где t – количество узлов на начальном слое; $t > 0; t \in \mathcal{N}$;

n – общее количество слоев; $n \geq 0; n \in \mathcal{N}$;

L – функция, определяющая количество связей между узлами двух слоев на заданном уровне;

G – функция, определяющая количество узлов следующего слоя.

Общее количество узлов определяется по формуле 2.

$$E(t,n,G) = \begin{cases} E(G(t,n), n-1, G) + G(t,n), & n > 0 \\ 0, & n = 0 \end{cases}, \quad (2)$$

где t – количество узлов на начальном слое; $t > 0; t \in \mathcal{N}$;

n – общее количество слоев; $n \geq 0; n \in \mathcal{N}$;

G – функция, определяющая количество узлов следующего слоя.

Крайним худшим случаем для поиска будет являться количество связей, которое необходимо проверить с самого верхнего слоя до самого нижнего, то есть обойти весь граф. Функция, описывающая данный крайний случай создает связи между каждым узлом начального слоя и каждым узлом следующего и имеет вид $L_{max} = (k, m, n) = km$. Тогда скорость роста числа связей $R(1, n, L_{max}, G_l)$ и количества узлов $E(1, n, G_l)$ на n уровнях будет иметь вид, представленный на рис. 2.

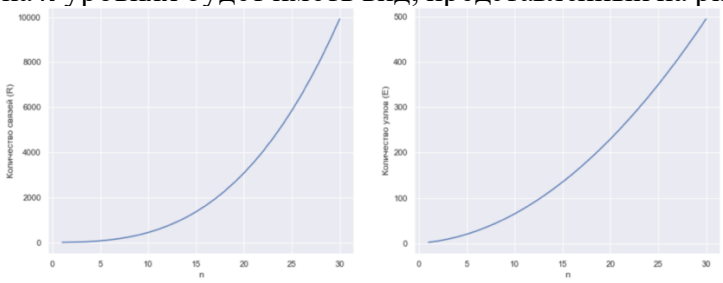

Рис. 2. Графики скорости роста числа связей и числа узлов при n числе слоев

Очевидно, что даже при небольшом количестве слоев $n \leq 30$, число узлов и связей довольно сильно различаются порядком, а сама зависимость далеко не линейна и ближе к степенной или экспоненциальной.

Помимо задачи непосредственно поиска, первостепенным является подготовка, обработка и загрузка данных в графовую базу, по которым и будет осуществляться сам поиск. Входными данными будет список умений, которыми обладают сотрудники. Они должны быть максимально конкретными для того, чтобы система смогла сама выявить нужные обобщения и классы, к которым данные умения относятся. Выходными данными являются сами умения, а также список обобщений, к которым данные умения относятся. Для формирования таких выходных данных, на первом этапе необходимо дополнить каждую входную компетенцию каким-либо ее определением или характеристикой Для этого используется сервис Wikipedia в котором находится нужная компетенция и берется её краткое описание посредством Python библиотеки «wikipedia». На втором этапе происходит разбиение краткого описания на словосочетания и выбор только словосочетаний, в котором имя существительное является вершиной, то есть главным словом, определяющим характеристику всей составляющей. После чего формируется список пар значений «компетенция – именная группа» который и загружается в графовую базу данных.

Поисковой запрос может включать в себя основные логические операции, такие, как И/ИЛИ, а также операцию группировки с приоритетом, а сам процесс поиска представляет собой обход по графу. Однако вначале необходимо преобразовать входящее выражение в дизъюнктивную нормальную форму (ДНФ), где в качестве логических литералов выступают компетенции сотрудников. ДНФ позволит преобразовать любое входящее логическое выражение в вид дизъюнкции конъюнкций литералов, что даст возможность разбить алгоритм на два этапа:

1) Поиск всех сотрудников, обладающих несколькими навыками одновременно:

– первый шаг– преобразование исходного выражения в ДНФ;

– следующие шаги – цикл по всем конъюнктивным группам и формирование запросов для обхода графа с учетом включения всех умений из группы.

2) Объединение всех результатов предыдущего этапа и удаление дубликатов.

Так же ДНФ позволит произвести минимизацию логического выражения, что ускорит алгоритм поиска, за счет уменьшения количества логических литералов.

Для поиска путей на графе используется так называемый двусторонний поиск в ширину.

Еще одним ограничением, которое стоит учитывать является сложность логического выражения, которое можно составить при формировании запроса поиска. Так как входящее выражение от пользователя может быть неоптимальное с точки зрения логики (лишние повторы и комбинации), то необходимо реализовать алгоритм минимизации логического выражения. Минимизация логического выражения является *NP*-полной задачей.

Разработанная веб-система является полностью кроссплатформенной и может быть запущена на любой операционной системе поддерживающей интерпретатор Python, а также программную платформу Node.js и графовую базу данных Neo4j.