Графы Знаний

План лекции:

4. Индуктивные знания
5. Создание и наполнение графа знаний

---

4. Индуктивные знания

В то время как дедуктивные знания характеризуется точными логическими следствиями, индуктивные знания включают обобщение моделей из заданного набора входных наблюдений, которые затем могут быть использованы для создания новых, но потенциально неточных предсказаний. Например, из данных схемы с географической и полетной информацией, мы можем наблюдать картину, что практически во всех столицах стран есть международные аэропорты, а значит, и предсказать, что если Сантьяго-это столица, то она, вероятно, имеет международный аэропорт, однако, прогнозы сделанные по этой модели не всегда срабатывают, например Вадуц, столица Лихтенштейна, не имеет (международного) аэропорта. Следовательно, предсказания часто будут связаны с уровнем доверия; например, мы можем сказать, что столица имеет международный аэропорт в 187 из 195 случаев, предлагая достоверность 0,959 для предсказаний, сделанных по этой модели. Затем мы называем знание, полученное индуктивно - индуктивным знанием, которое включает в себя как модели, используемые для кодирования паттернов, так и предсказания, сделанные этими моделями. Хотя индуктивное знание может быть ошибочным, оно может быть очень ценным. На рис. 23 представлен обзор индуктивных методов, обычно применяемых к графам знаний. В случае неконтролируемых методов существует богатый объем работ по графовой аналитике, которая использует хорошо известные функции/алгоритмы для обнаружения сообществ или кластеров, поиска центральных узлов и ребер и т. д. В качестве альтернативы графа знаний может использовать самонаблюдение для изучения низкоразмерной числовой модели графа знаний, которая (как правило) сопоставляет входные ребра с выходным показателем правдоподобия, указывающим вероятность того, что ребро истинно. Структура графов также может быть непосредственно использована для контролируемого обучения, как это исследуется в контексте графовых нейронных сетей. Наконец, в то время как вышеупомянутые методы изучают числовые модели, символьное обучение может изучать символьные модели – то есть логические формулы в форме правил или аксиом графа самоконтролируемым образом. Теперь мы по очереди обсудим каждую из вышеупомянутых техник.

Графовая Аналитика

Аналитика-это процесс обнаружения, интерпретации и передачи значимых паттернов , присущих (как правило, большим) коллекциям данных. Графовая аналитика-это применение аналитических процессов к (как правило, большим) графовым данным. Природа графов, естественно, поддается определенным типам анализа, которые делают выводы об узлах и ребрах на основе топологии графа, т. е. таким образом, графовая аналитика черпает многие из своих методов из смежных областей, таких как теория графов и сетевого анализа, которые использовались для изучения графов, представляющих социальные сети, интернет–маршрутизацию, транспортные сети, экосистемы, белково-белковые взаимодействия, лингвистические взаимодействия и многое другое. Возвращаясь к домену в нашем примере, туристическая доска может использовать графовую аналитику для извлечения знания о, например: транспортно-пересадочных узлах, которые обслуживают множество туристических достопримечательностей (роль); группировки достопримечательностей, которые посещают одни и те же туристы (общий обнаружения);

достопримечательности, которые могут стать недоступными в случае забастовок или других факторов (подключения), либо пар достопримечательностей, которые аналогичны друг другу (узел сходство). Учитывая, что такая аналитика потребует сложного крупномасштабного графа, для иллюстрации, на рис. 24 мы приведем более краткий пример некоторых транспортных связей в Чили, направленных к популярным туристическим точкам. Сначала мы представим набор ключевых методов, которые могут быть применены для графовой аналитики. Затем мы обсудим фреймворки и языки, которые могут быть использованы для вычисления такой аналитики на практике. Учитывая, что многие традиционные графовые алгоритмы определены для неатрибутированных графов, мы затем опишем способы, с помощью которых аналитика может быть применена к ориентированным графам с именованными ребрами. Наконец, мы обсудим потенциальные связи между графовой аналитикой, запросами и рассуждениями.

Методы. Для графовой аналитики можно применять самые разнообразные методы. Далее мы перечислим некоторые из основных методов – как признанные.

- ***Обнаружение центров*****:** цель состоит в том, чтобы определить наиболее важные (или центральные) узлы или ребра графа. Конкретные показатели центральности узлов включают степень, промежуточность, близость, собственный вектор, PageRank, HITS, Katz и другие. Центральность между ними также может быть применена к узлам. Мера центральности узлов позволила бы, например, предсказать транспортные узлы на рис. 24.
- ***Обнаружение сообществ:*** цель состоит в том, чтобы идентифицировать сообщества в графе, то есть подграфы, которые более плотно связаны внутри, чем с остальной частью графа. Алгоритмы обнаружения сообществ - например, алгоритмы минимального разреза, распространения меток, модульность Лувена и т.д. Обнаружение сообщества, примененно к рис. 24, может, например, обнаружение сообщества слева (относится к северу чили), справа (относится к югу чили), а возможно, и центр (относится к городам с аэропортами).
- ***Связность:*** цель состоит в том, чтобы оценить, насколько хорошо связан граф, выявляя, например, устойчивость и (недостижимость)достижимость элементов графа. Конкретные методы включают измерение плотности графа или k-связности, обнаружение сильно связанных компонентов и слабо связанных компонентов и т. д. В контексте рисунка 24 такой анализ может сказать нам, что маршруты В Osorno Volcano и Piedras Rojas являются наиболее “хрупкими”, отсоединяясь от основных узлов, если один из двух автобусных маршрутов останавливается.
- ***Сходство узлов:*** цель состоит в том, чтобы найти узлы, которые похожи на другие узлы в силу того, как они связаны в пределах их окрестности. Метрики подобия узлов могут быть вычислены с использованием структурной эквивалентности, случайных блужданий, диффузионных ядер и т. д. Эти методы дают понимание того, что связывает узлы, и, следовательно, в чем они похожи. В контексте рисунка 24 такой анализ может сказать нам, что Calama и Arica являются подобными узлами, основанными как на наличии обратных рейсов Santiago , так и на обратных автобусах San Pedro .

В то время как предыдущие методы принимают только граф в качестве входных данных, другие формы графовой аналитики могут дополнительно принимать узел, пару узлов и т. д.

- ***Поиск пути:*** цель состоит в том, чтобы найти пути в графе, как правило, между парами узлов, заданных в качестве входных данных. Существуют различные технические определения, ограничивающие набор допустимых путей между такими узлами, включая простые пути, которые не посещают один и тот же узел дважды, самые короткие пути, которые посещают наименьшее количество ребер, или – как уже обсуждалось ранее – регулярные запросы пути, которые ограничивают метки ребер, которые могут быть

пройдены путем. Мы могли бы использовать такие алгоритмы, чтобы найти, например, кратчайший путь(ы) на рис. 24 от Torres del Paine до Moon Valley.

Большинство таких методов было предложено и изучено для простых графов или ориентированных графов без меток ребер. Мы обсудим их применение к более сложным графовым моделям – и то, как они могут сочетаться с другими методами.

Фреймворки. Были предложены различные фреймворки для крупномасштабной Графовой аналитики, часто в распределенной (кластерной) среде. Среди них можно упомянуть Apache Spark (GraphX), GraphLab, Pregel, Signal–Collect, Shark и др. Эти графопараллельные фреймворки применяют систолическую абстракцию, основанную на ориентированном графе, где узлы являются процессорами, которые могут посылать сообщения другим узлам вдоль ребер. Далее вычисление является итерационным, где в каждой итерации каждый узел считывает сообщения, полученные через внутренние ребра (и, возможно, свое собственное предыдущее состояние), выполняет вычисление, а затем отправляет сообщения через внешние ребра на основе результата вычислений. Затем эти фреймворки определяют систолическую вычислительную абстракцию поверх обрабатываемого графа данных: узлы и ребра в графе данных становятся узлами и ребрами в систолическом графе. Для примера предположим, что мы хотим вычислить места, до которых наиболее (или наименее) легко добраться маршрутами, показанными на графе на рис.24. Хороший способ измерить это - использовать центральность, где мы выбираем PageRank, который вычисляет вероятность того, что турист случайно пойдет по маршрутам, показанным на графе, оказавшись в определенном месте после заданного числа “прыжков”. Мы можем реализовать PageRank на больших графах, используя параллельную структуру графов. На рис. 25 мы приводим пример итерации PageRank для иллюстративного подграфа на рис. 24. Узлы инициализируются со счетом 1 $|V|$ = 1, где мы предполагаем, что турист имеет равные шансы стартовать в любой точке. В сообщении этапа, каждый узел v проходит счетом $dpi\,(v)\,|E(v)|$ на каждого из исходящих ребер, где мы обозначим через d - постоянный коэффициент демпфирования, который используется для обеспечения сходимости (как правило, d = 0.85, указывающих на вероятность того, что турист хаотично “прыгает” до любого места), на Pi (v) счет узла v в итерации i (вероятность, что турист, находясь на узле v после i шага), и $|E(v)|$ количество исходящих ребер v. Агрегирование фаз (agg) для v затем суммирует все входящие сообщения, полученные вместе с постоянной долей коэффициента демпфирования ($1-d\,|V|$), чтобы вычислить Pi+1 (v). Затем мы переходим на фазу следующих итераций, продолжающиеся до прекращения некоторым критерием (напр., числа итераций или остаточной порог, и т. д.) и финальные результаты выводятся. Хотя приведенный пример является для PageRank, систолической абстракции достаточно, чтобы поддерживать широкое разнообразие графовой аналитики, в том числе и ранее упомянутых. Алгоритм в этой структуре состоит из функций для вычисления значений сообщений в фазе сообщений (Msg) и для накопления сообщений в фазе агрегации (Agg). Фреймворк позаботится о распространении, передаче сообщений, отказоустойчивости и т. д. Однако такие фремворки – на основании сообщений, пересылаемых между соседями – имеют ряд ограничений: не все типы анализа могут быть реализованы в таких фреймворках. Поэтому следует разрешить дополнительные функции, такие как глобальный шаг, который выполняет глобальные вычисления на всех узлах, что делает результат доступным для каждого узла; или мутационный шаг, что позволяет добавление или удаление вершин и ребер во время обработки.

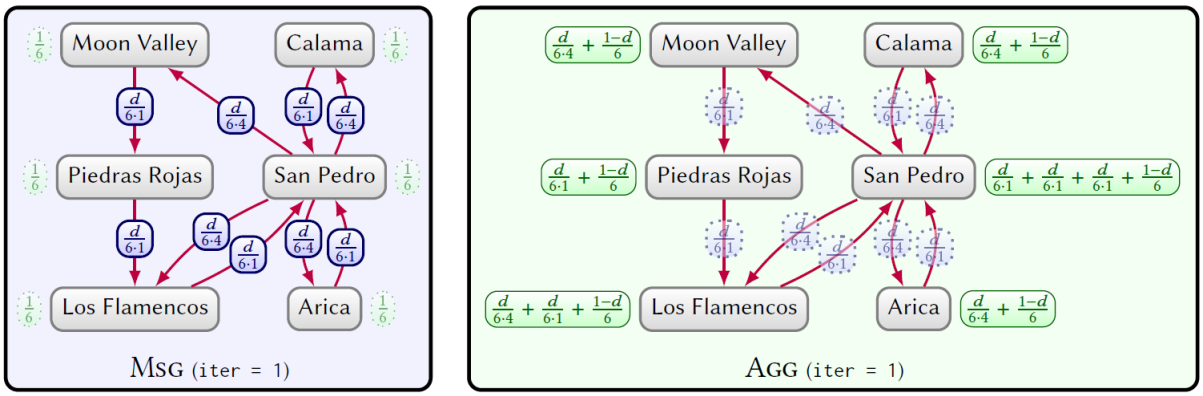


Рис. 25. пример систолической итерации PageRank для выборочного подграфа рис. 24

Аналитика на графах данных. Как уже упоминалось, большинство аналитик, представленных до сих пор, в своей “родной” форме применимы к неориентированным или ориентированным графам без граничных метаданных-то есть меток ребер или пар свойств – значений, типичных для моделей графовых данных. Ряд стратегий может быть применен для того, чтобы сделать графы данных предметом анализа этой формы:

- **Проекция** включает в себя простое “проецирование” неориентированного или направленного графа путем выбора подграфа из графа данных, из которого отбрасываются все метаданные ребер; например, рисунок 24 может быть результатом извлечения подграфа, вызванного шиной меток ребер, и преобразования в более крупный граф данных, где метки затем отбрасываются для создания направленного графа.
- **Взвешивание** включает в себя преобразование метаданных ребер в числовые значения в соответствии с некоторой функцией. Многие из вышеупомянутых методов, легко адаптируются к взвешенным графам; например, мы могли бы рассмотреть веса на графе рис. 24, обозначающие продолжительность поездки (или цене, трафик и т. д.), а затем вычислить кратчайшие пути, складывая длительности каждого маршрута.
- **Преобразование** включает в себя преобразование графа в модель более низкой размерности. Преобразование может быть с потерями, что означает, что исходный граф не может быть восстановлен; или без потерь, что означает, что исходный граф может быть восстановлен. На рис. 26 представлен пример преобразования с потерями и без потерь из направленного графа в направленный граф. В преобразовании с потерями мы не можем сказать, например, содержал ли исходный граф ребро Iquique –flight→ Santiago или Iquique –flight→ Arica т. д. Преобразование без потерь должно вводить новые узлы (похожие на овеществление) для сохранения информации о направленных помеченных ребрах. Оба преобразованных графа в дальнейшем пытаются сохранить направленность исходного графа.
- **Настройка** включает в себя изменение аналитической процедуры для включения метаданных ребер, например, в случае поиска пути на основе выражений пути. Другие примеры могут включать структурные меры сходства узлов, которые учитывают не только общих соседей, но и общих соседей, связанных ребрами с одной и той же меткой, или агрегатные меры центральности, которые учитывают важность ребер, сгруппированных по метке, и т. д.

Результаты аналитического процесса могут кардинально меняться в зависимости от того, какая из предыдущих стратегий выбрана для подготовки данных к анализу. Этот выбор может быть нетривиальным, чтобы сделать его, может потребоваться эмпирическая проверка. Необходимы

дополнительные исследования для более общего понимания влияния таких стратегий на результаты различных аналитических методов.

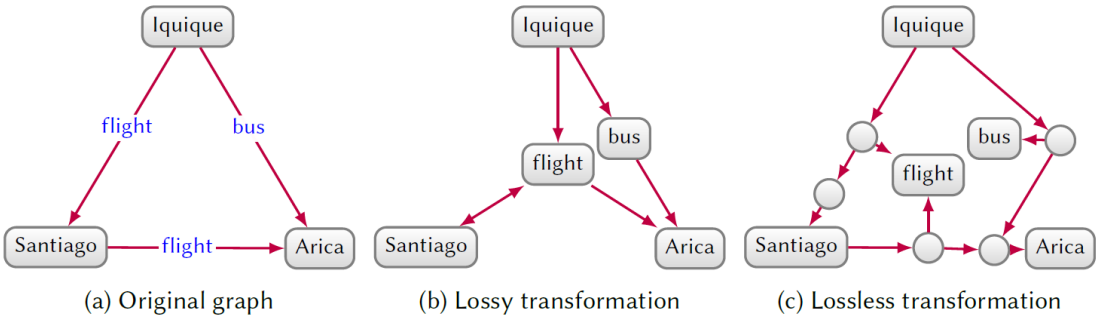


Рис. 26. Преобразования из ориентированного краевого графа в ориентированный граф

Аналитика с запросами. Были предложены различные языки для запроса к графам. Можно рассмотреть множество способов, которыми языки запросов и аналитика могут дополнять друг друга. Во-первых, мы можем рассмотреть возможность использования языков запросов для проектирования или преобразования графа, подходящего для конкретной аналитической задачи, например для извлечения графа на рис.24 из более крупного графа данных. Языки запросов, такие как SPARQL, Cypher и G-CORE, позволяют получать графы, где такие запросы могут использоваться для выбора подграфов для анализа. Эти языки также могут выражать некоторую ограниченную (нерекурсивную) аналитику, где агрегации могут использоваться, например, для вычисления степени центральности; они также могут иметь некоторую встроенную аналитическую поддержку, где, например, Cypher позволяет находить кратчайшие пути. С другой стороны, аналитика может внести свой вклад в процесс запроса с точки зрения оптимизации, где , например, анализ связности может предложить, как лучше распределить большой граф данных по нескольким серверам для запроса, используя, например, минимальные сокращения. Аналитика также используется для ранжирования результатов запросов по большим графам, выбирая наиболее важные результаты для представления пользователю. Например, из полного графа данных, собранного туристическим советом, рассмотрим предстоящую забастовку авиакомпаний, где совет хочет найти события во время забастовки с местами проведения в городах, недоступных из Сантьяго общественным транспортом из-за забастовки. Гипотетически мы могли бы использовать запрос для извлечения транспортной сети, кроме авиакомпаний (предположим, на рисунке 3, что авиакомпания информация доступна), использование аналитики для извлечения сильно связных компонентов содержащих Сантьяго, и, наконец, использовать запрос, чтобы найти события в городах, не в Сантьяго на указанные даты. В то время как можно было бы решить эту задачу с помощью императивного языка, такого как Gremlin , GraphX или R, более декларативные языки также исследуются для более легкого выражения таких задач, с предложениями, включающими расширение графовых языков запросов с рекурсивными возможностями, объединение линейной алгебры с реляционной (запросной) алгеброй и т. д.

Аналитика с притяжением. Графы знаний часто ассоциируются с семантической схемой или онтологией, которая определяет семантику терминов предметной области, порождая притяжение. Применение аналитики с такими притяжениями или без них – например, до или после материализации – может привести к радикально различным результатам. Например, обратите внимание, что ребро Santa Lucía –hosts→ EID15 семантически эквивалентно ребру EID15 –venue→ Santa Lucía , если hosts –inv. of→ venue вызывается обратная аксиома; однако эти ребра далеки от эквивалентности с точки зрения аналитических методов, которые рассматривают направление ребра, для которого включение одного типа ребра, или другого, или обоих, может

иметь большое влияние на конечные результаты. Насколько нам известно, сочетание аналитики и влечения не было хорошо изучено, оставляя открытыми интересные исследовательские вопросы. Идя вдоль этих линий, она может быть интересна для изучения семантически инвариантных аналитик, которые дают одинаковые результаты более семантически эквивалентных графов (т. е. графов, которые влекут за собой друг друга), таким образом анализ смыслового содержания знаний графа, а не просто топологические особенности графических данных.

Встраивание Графа Знаний (Embedding)

Методы машинного обучения получили значительное развитие в последние годы. В контексте графов знаний машинное обучение может быть использовано либо для непосредственного уточнения графа знаний, либо для последующих задач, использующих граф знаний, таких как рекомендации, извлечение информации, ответы на вопросы, релаксация запросов, аппроксимация запросов и т.д. Однако многие традиционные методы машинного обучения предполагают плотные числовые входные представления в виде векторов, что весьма отличается от того, как обычно выражаются графами. Как графы, или их узлы, ребра и т. д. – могут быть представлены в виде числовых векторов? Для первой попытки представления графа с помощью векторов можно использовать генерацию вектора для каждого узла длиной $|L| \cdot |V|$ – с $|V|$ количеством вершин входного графа и $|L|$ количеством меток ребер – поставив единицу на соответствующий индекс, чтобы указать на наличие соответствующего ребра в графе, или ноль в противном случае. Однако такое представление, как правило, приводит к большим и разреженным векторам, что пагубно сказывается на большинстве моделей машинного обучения. Основная цель методов встраивания графа знаний состоит в создании плотного представления графа (т. е. встраивания графа) в непрерывное низкоразмерное векторное пространство, которое затем может быть использовано для задач машинного обучения. Размерность d вложения фиксирована и обычно низка (часто, например, $50 \geq d \geq 1000$). Обычно встраивание графа состоит из вложения сущности для каждого узла: вектора с d измерениями, которые мы обозначаем через e; и вложения отношения для каждой метки ребра: (обычно) вектора с d измерениями, которые мы обозначаем через r. Общая цель этих векторов состоит в том, чтобы абстрагировать и сохранить скрытые структуры в графе. Существует много способов, с помощью которых это понятие встривания может быть реализовано. Чаще всего, учитывая ребро s p o, конкретный подход к встраиванию определяет скоринговую функцию, которая принимает e_s (встраивание сущности узла s), r_p (встраивание сущности метки ребра p) и e_o (встраивание сущности узла o) и вычисляет правдоподобие ребра: насколько вероятно, что оно истинно. Полученные вложения затем можно рассматривать как модели, изучаемые с помощью самонаблюдения, которые кодируют (латентные) особенности графа, сопоставляя входные ребра с выходными оценками правдоподобия.

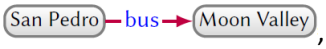
Затем вложения могут быть использованы для ряда низкоуровневых задач, связанных с узлами и метками ребер графа, из которого они были вычислены. Во-первых, мы можем использовать функцию оценки правдоподобия для присвоения уровня правдоподобности ребрам, которые, например, могут быть извлечены из внешнего источника. Во-вторых, функция достоверности может быть использована для завершения ребра с отсутствующими метками для целей прогнозирования; например, на рис. 24, мы могли бы спросить, какие узлы в графе, скорее всего, для завершения ребра 

, где – кроме того , что есть 

, что уже дано – мы могли бы интуитивно ожидать 

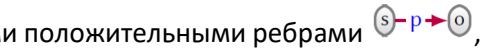
, что тоже может быть вероятным. В-третьих, модели встраивания обычно присваивают сходные векторы сходным узлам и сходным меткам ребер, и таким образом они могут использоваться в качестве основы мер сходства, которые могут быть полезны для поиска дубликатов узлов, ссылающихся на одну и ту же сущность, или для целей предоставления рекомендаций. Был предложен широкий спектр методов встраивания графов знаний, где цель состоит в том, чтобы обеспечить высокоуровневое введение в некоторые из наиболее популярных методов, предложенных до сих пор. Сначала мы обсудим трансляционные модели, которые

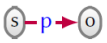
принимают геометрическую форму, посредством которой вложения отношений переводят субъектные сущности в объектные сущности в низкоразмерном пространстве. Затем мы опишем модели тензорной декомпозиции, которые извлекают скрытые факторы, аппроксимирующие структуру графа . После этого мы обсудим нейронные модели, которые используют нейронные сети для обучения вложений, которые обеспечивают точные оценки правдоподобия. Наконец, мы обсуждим лексические модели, которые используют существующие методы встраивания слов, предлагая способы генерации графоподобных аналогов для их ожидаемых (текстовых) входных данных.

Трансляционные модели. Трансляционные модели интерпретируют метки ребер как преобразования от узлов субъекта (он же источник или голова) к узлам объекта (он же цель или хвост); например, в 

, ребре метка ребра рассматривается как преобразование 

к 

для других ребер аналогично. Наиболее элементарным подходом в этом семействе является TransE. Над всеми положительными ребрами 

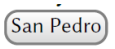
, TransE изучает векторы e_s, r_p и e_o, стремясь сделать $e_s + r_p$ как можно ближе к e_o. И наоборот, если близость является отрицательным примером, TransE пытается изучить представление, которое удерживает $e_s + r_p$ вдали от e_o. Для иллюстрации на рис. 27 приведен игрушечный пример двумерных ($d = 2$) вложений сущностей и отношений, вычисленных TransE. Мы сохраняем ориентацию векторов аналогичной исходному графу для ясности. Для любого ребра 

в исходном графе добавление векторов $e_s + r_p$ должно аппроксимировать e_o. В этом игрушечном примере векторы точно соответствуют тому, где, например, добавление векторов для 

(eL.) и к западу от (rwo.) дает вектор, соответствующий 

(eC.). Мы можем использовать эти вложения для предсказания ребер (среди других задач); например, для того, чтобы предсказать, какой узел в графе, скорее всего, будет к западу от 

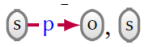
(A.), вычисляя eA. + rwo. мы видим, что результирующий вектор (пунктирная линия на рис. 27C) находится ближе к восточному, таким образом, прогнозирование 

(т.) является наиболее вероятным такого узла. Кроме этого, TransE может быть слишком упрощенным, например, на рис. 24, автобус не только превращает 

в 

а также с 

, 

,и так далее. TransE в этом случае будет стремиться дать одинаковые векторы всем таким целевым местоположениям, что может быть невозможно при наличии других ребер. TransE также будет стремиться присвоить циклическим отношениям нулевой вектор, поскольку направленные компоненты будут стремиться компенсировать друг друга. Для решения таких проблем было исследовано множество вариантов ТрансЕ. Среди них, например, TransH представляет различные отношения с использованием различных гиперплоскостей, где ребро 

,

сначала проецируется на гиперплоскость p, до того, как будет изучен перевод (без влияния ребер с другими метками for 

и for 

). TransR обобщает этот подход, проецируя 

и 

в векторное пространство, специфичное для p, которое включает в себя умножение вложений сущностей для 

и 

на проекционную матрицу, специфичную для p. TransD упрощает TransR, связывая сущности и отношения со вторым вектором, где эти вторичные векторы используются для проецирования сущности в специфичное для отношений векторное пространство. И наконец, RotatE предлагает трансляционные вложения в сложном пространстве, что позволяет захватить больше характеристик отношений, таких как направление, симметрия, инверсия, антисимметрия и композиция. Вложения также были предложены в неевклидовом пространстве, например, MuRP использует вложения отношений, которые преобразуют вложения сущностей в гиперболическое пространство Poincaré, кривизна которого обеспечивает больше “пространства” для разделения сущностей по отношению к размерности.

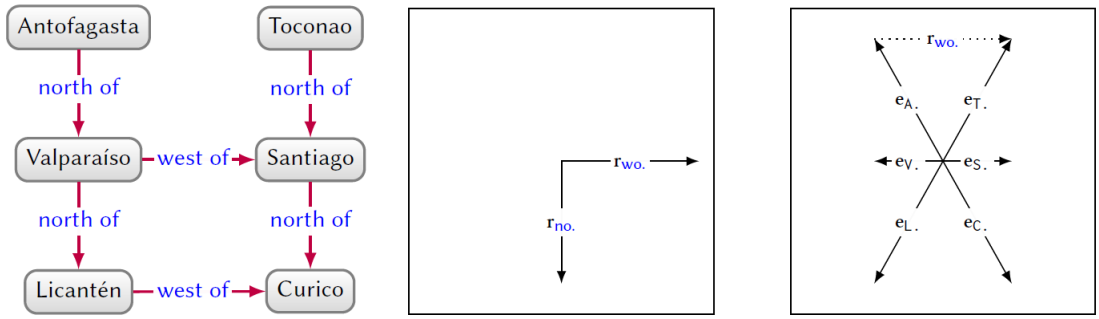


Рис. 27. Игрушечный пример двумерных отношений и вложений сущностей, изученных TransE; вложения сущностей используют аббревиатуры и включают пример векторного сложения, чтобы предсказать, что находится к западу от Антофагасты

Модели тензорной декомпозиции. Второй подход к получению вложений графов заключается в применении методов, основанных на тензорной декомпозиции. Тензор-это многомерное числовое поле, которое обобщает скаляры (тензоры 0-го порядка), векторы (тензоры 1-го порядка) и матрицы (тензоры 2-го порядка) на произвольную размерность/порядок. Тензоры стали широко используемой абстракцией для машинного обучения. Декомпозиция тензора включает в себя разложение тензора на более “элементарные” тензоры (например, более низкого порядка), из которых исходный тензор может быть перекомпозирован (или аппроксимирован) фиксированной последовательностью основных операций. Эти элементарные тензоры можно рассматривать как фиксирующие скрытые факторы, лежащие в основе информации, содержащейся в исходном тензоре. Существует много подходов к тензорной декомпозиции, где мы теперь кратко представим основные идеи, лежащие в основе ранговых декомпозиций. Оставляя в стороне графы, рассмотрим (a, b)-матрицы (то есть порядок-2 тензора) C, где a - количество городов в Чили, b - число месяцев в году, и каждый элемент $(c)_{ij}$ обозначает среднюю температуру i- го города B j - й месяц. Отметить, что Чили представляет собой длинную, тонкую страну – от приполярного климата на юге к пустынному на севере. Мы можем найти разложение C на два вектора, представляющие латентные факторы, в частности X (с a элементов) дает более низкие значения для городов на более низкой широте, и Y (с b элементов), давая более низких значений в течение нескольких месяцев при более низких температурах. B (маловероятном) случае, если существуют векторы X и y, таких, что C является именно внешним произведением двух векторов вида (x ⊗ y B = C) мы называем с рангом-1 матрицы; мы сможет тогда точно закодировать с помощью $a + b$ меры, а не $a \times b$. Однако в большинстве случаев, чтобы получить точно C, нам нужно будет суммировать несколько матриц ранга 1, где ранг r C - это минимальное число матриц ранга 1, которые необходимо суммировать, чтобы получить точно C, так что x1 ⊗ y1+. .. xr ⊗ yr = C. B Примере с температурой x2 ⊗ y2 может соответствовать поправке на значение, x3 ⊗ y3-более высокой дисперсии температуры дальше на юг и т. д. A (Low) ранг разложение матрицы a затем устанавливает предел d по рангу и вычисляет векторы (X1, У1, . . . , xd, гd), таких, что X1 ⊗ У1 + . . . + xd ⊗ гd дает лучшие d-ранг аппроксимации C. Отметив, что для создания тензоров n-порядка, нам нужно вычислить внешний продукт n векторов, мы можем обобщить эту идею в сторону низких рейтингов разложения тензоров; этот метод называется каноническим полиадическим (КП) разложением. Например, у нас есть тензор третьего порядка c, содержащий месячные температуры для чилийских городов в четыре разных времени суток, которые могут быть аппроксимированы с X1 ⊗ У1 ⊗ Z1 в + . . . xd ⊗ гd ⊗ зd (например, X1 может быть широтный фактор, У1 ежемесячный варьирования этого фактора, и Z1 ежедневно варьирования этого фактора, и так далее). Затем существуют различные алгоритмы для вычисления (приближенных) КП-разложений, включая чередующиеся наименьшие квадраты, алгоритм Дженнриха и тензорный степенной метод.

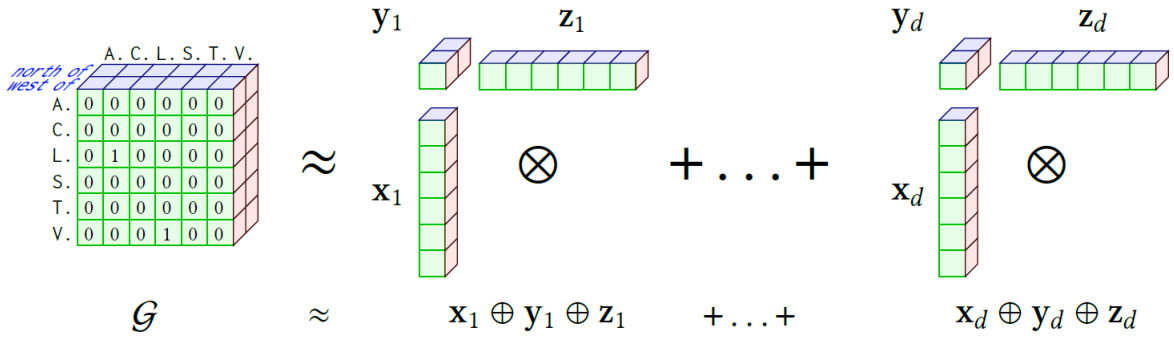


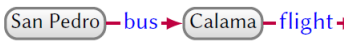
Рис. 28. Абстрактная иллюстрация СР d-рангового разложения тензора, представляющего граф на рис. 27а

Возвращаясь к графам, аналогичные принципы могут быть использованы для разложения графа на векторы, что приводит к вложениям. В частности, такой граф может быть закодирован как один-тензор 3-го порядка G с $|V| \times |L| \times |V|$ элементов с элементами $(г)_{ijk}$ со значением 1 для i- го узла, k- й связи с j - й меткой, или ноль в противном случае. Как уже упоминалось ранее, такой тензор обычно будет очень большим и разреженным, где, таким образом, применимы ранговые разложения. КП разложения может вычислить последовательность вектора ($Х_1$, $У_1$, Z_1, так . . . , $х_d$, $г_d$, $з_d$), таких, что $Х_1 \otimes У_1 \otimes Z_1$ в + . . . + $х_d \otimes Г_d \otimes З_d \approx$ г. Проиллюстрируем эту схему на рис. 28. Пусть X, Y, Z обозначают матрицы , образованные $[\mathbf{x}_1 \cdots \mathbf{x}_d], [\mathbf{y}_1 \cdots \mathbf{y}_d], [\mathbf{z}_1 \cdots \mathbf{z}_d]$ соответственно, причем каждый вектор образует столбец соответствующей матрицы, тогда мы могли бы извлечь i-ю строку Y как вложение для i -го отношения, а j - ю строки X и Z как два вложения для j - й сущности. Однако вложения графа знаний обычно направлены на то, чтобы присвоить каждому объекту один вектор. DistMult - способ для вычислений знания графового вложения в зависимости от уровня разложения, где каждая сущность и соотношение связано с вектором размерности d, такие , что для ребра (s)–p→(o), функция достоверности $\sum_{i=1}^{d} (\mathbf{e}_s)_i (\mathbf{r}_p)_i (\mathbf{e}_o)_i$ определена, где $(эс)_i$, $(рп)_i$ и $(эо)_i$ обозначают i- й элемент векторов ЭП, РП, ЭО, соответственно. Таким образом, цель состоит в том, чтобы изучить векторы для каждого узла и метки ребра, которые максимизируют правдоподобие положительных ребер и минимизируют правдоподобие отрицательных ребер. Такой подход приравнивается к КП декомпозиции графа G, но где субъекты имеют один вектор, который используется дважды: $Х_1 \otimes У_1 \otimes Х_1$ + . . . + $х_d \otimes г_d \otimes х_d \approx$ г. слабость такого подхода заключается зв том, что значения функции правдоподобия для (s)–p→(o) всегда будут равны значениям для (o)–p→(s) ; иначе говоря, DistMult не считает ребра направлении. Вместо того чтобы использовать вектор в качестве вложения отношения, RESCAL использует матрицу, которая позволяет комбинировать значения из es и eo во всех измерениях и, таким образом, может захватывать (например) направление ребра. Однако RESCAL имеет более высокую сложность с точки зрения размера и времени, чем DistMult. HolE использует векторы для вложений отношений и сущностей, но предлагает использовать оператор круговой корреляции, который принимает суммы по диагоналям внешнего произведения двух векторов, чтобы объединить их. Этот оператор не является коммутативным и, таким образом, может учитывать направление ребра. Комплекс, с другой стороны, использует комплексный вектор (т. е. вектор, содержащий комплексные числа) в качестве реляционного вложения, что аналогично позволяет нарушить вышеупомянутую симметрию DistMult's скоринговой функции DistMult при сохранении низкого числа параметров. SimplE предлагает вычислить стандартную декомпозицию КП, вычисляющую два начальных вектора для сущностей из X и Z, а затем усредняющую члены по X, Y, Z для вычисления конечных оценок правдоподобия. Tacer использует другой тип декомпозиции, называемый декомпозицией Такера, который вычисляет меньший тензор "ядра" T и последовательность из трех матриц A, B и C,

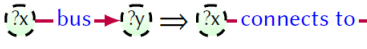
таких что G ≈ T ⊗ A ⊗ B ⊗ C – где вложения сущностей взяты из A и C, а вложения отношений взяты из B. Из этих подходов Такер в настоящее время предоставляет современные результаты по стандартным критериям.

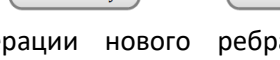
Нейронные модели. Ограничение ранее рассмотренных подходов заключается в том, что они предполагают либо линейные (с сохранением сложения и скалярного умножения), либо билинейные (например, матричное умножение) операции над вложениями для вычисления оценок правдоподобия. Ряд подходов скорее используют нейронные сети для изучения вложений с нелинейными скоринговыми функциями для правдоподобия. Одним из первых предложений нейронной модели семантического сопоставления (MCC), который узнает параметры (как вес: w, w') две функции fw(es, rp) и gw' (eo, rp) – такие , что скалярное произведение в результате для обеих функций fw(es, rp) · gw' (eo, rp) – обеспечивает правдоподобность результата. Предложены как линейный, так и билинейный варианты fw и gw'. Еще раньше предложены были тензоры нейронных сетей (THC), которые предлагают поддержку внутренних весов, таких, что правдоподобность результат вычисляется по сложной функции, которая сочетает в себе внешнее произведение es ⊗W ⊗ eo со стандартным нейронным слоем над эс и эо, которые, в свою очередь, сочетается с КП, для создания правдоподобного результата. Использование тензора приводит к большому количеству параметров, что ограничивает масштабируемость. Многослойный персептрон (МП) - это более простая модель, в которой es, rp и eo объединяются и подаются в скрытый слой для вычисления оценки правдоподобия. Ряд более поздних подходов предложил использовать сверточные ядра в своих моделях. ConvE предлагает создать матрицу из es и rp путем “обертывания” каждого вектора по нескольким строкам и объединения обеих матриц. Объединенная матрица служит входным сигналом для набора (2D) сверточных слоев, который возвращает тензор карты объектов. Тензор карты объектов векторизуется и проецируется в d мерный вектор с помощью параметризованного линейного преобразования. Оценка правдоподобия затем вычисляется на основе точечного произведения этого вектора и eo. Недостатком ConvE является то, что, обертывая векторы в матрицы, он накладывает искусственную двумерную структуру на вложения. HypER - аналогичная модель, использующая свертки, но избегающая необходимости оборачивать векторы в матрицы. Вместо этого, полностью связанный слой (называемый “гиперсетью”) применяется к rp и используется для создания матрицы специфичных для отношений сверточных фильтров. Эти фильтры применяются непосредственно к es, чтобы дать карту объектов, которая векторизуется. Затем применяется тот же процесс, что и в ConvE: результирующий вектор проецируется в d-мерный вектор, а точечное произведение применяется с eo для получения оценки правдоподобия. Показано, что полученная модель превосходит ConvE по стандартным бенчмаркам. Представленные подходы давят различными балансами с точки зрения выразительности и количества параметров, которым нужно обучать. В то время как более выразительной модели, например, HTH, возможно, лучше подходят для более сложных функций правдоподобия с низкой размерностью вложения с помощью нескольких скрытых параметров, более простых моделей и сверточных сетей, которые позволяют обмен параметрами, применяя те же (как правило, малые) ядра из различных регионов матрицы, требуя меньше обработки параметров в целом и являются более масштабируемыми.

Лингвистические модели. Методы встраивания были впервые исследованы как способ представления естественного языка в рамках машинного обучения, причем word2vec и GloVe были двумя основополагающими подходами. Оба подхода вычисляют вложения для слов, основанных на больших блоках текста, так что слова, используемые в сходных контекстах (например, “лягушка”, “жаба”), имеют сходные векторы. Word2vec использует нейронные сети, обученные либо предсказывать текущее слово из окружающих слов (непрерывный мешок слов), либо предсказывать окружающие слова с учетом текущего слова (непрерывная скипграмма). GloVe скорее применяет регрессионную модель к матрице вероятностей совпадения пар слов. Вложения, генерируемые обоими подходами, широко используются в задачах обработки естественного языка.

Таким образом, другой подход к вложениям графов заключается в использовании проверенных подходов к языковым вложениям. Однако если граф состоит из неупорядоченного набора последовательностей триплетов (то есть множества ребер), то текст на естественном языке состоит из последовательностей членов произвольной длины (то есть предложений слов). Таким образом, RDF2Vec выполняет случайные блуждания по графу и записывает пути (последовательность пройденных узлов и меток ребер) в виде “предложений”, которые затем подаются в качестве входных данных в модель word2vec. Примером такого пути, извлеченного из рисунка 24, может служить, например, 

тот случай, когда в статье исследуется 500 путей длиной 8 на объект. RDF2Vec также предлагает второй режим, в котором последовательности генерируются для узлов из канонически помеченных поддеревьев, корневым узлом которых они являются, где статья экспериментирует с поддеревьями глубины 1 и 2, и наоборот, KGloVe основана на модели GloVe. Подобно тому, как оригинальная GloVe модель рассматривает слова, которые часто встречаются в текстовых окнах, как более связанные, KGloVe использует персонализированный PageRank для определения наиболее связанных узлов с данным узлом, результаты которого затем передаются в модель GloVe.

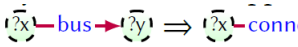
Модели, совместимые с последствиями. Вложения до сих пор рассматривали только для графа данных. Но что делать, если существует онтология или набор правил? Такое дедуктивное знание может быть использовано для улучшения вложений. Одним из подходов является использование правил ограничения для уточнения прогнозов вложений; например, некоторые авторы используют функциональные и обратные функциональные определения как ограничения, такие, что, например, если мы определим, что событие может иметь более одного значения результата, это используется для уменьшения вероятности ребер, что бы назначить несколько результатов на событие. Более поздние подходы предлагают объединенные вложения, которые объединяют граф данных и правила для вычисления вложения. Возможно вычислять вложения сущностей и отношений с использованием трансляционной модели (в частности, TransE), адаптированной для дальнейшего рассмотрения правил с использованием t-норм нечеткой логики. На рис. 24 рассмотрим простое правило 

, которое мы можем использовать для присвоения оценок правдоподобия новым ребрам, например $e1$: 

мы можем далее применить предыдущее правило для генерации нового ребра $e2$: 

из предсказанного ребра $e1$. Но какое правдоподобие мы должны приписать этому второму ребру? Если $p1$ и $p2$ - текущие оценки правдоподобия $e1$ и $e2$ (инициализированные с использованием стандартного вложения), то нечеткая логика t-нормы предполагает, что правдоподобие обновляется следующим образом $p1p2 - p1 + 1$. Затем вложения обучаются совместно присваивать более высокие значения правдоподобия положительным примерам по сравнению с отрицательными примерами как ребер, так и основных правил. Примером положительного основного правила, основанного на рис.24, могут быть 

. отрицательные основные правила, случайным образом заменяющие отношение в главе правила; например, 

. Другие авторы используют совместную модель над основными правилами (возможно, часто правила с доверительными баллами) и баллами правдоподобия, чтобы выровнять обе формы оценки для невидимых ребер. Выработка основных правил может быть дорогостоящей. Альтернативный подход, называемый FSL, замечает, что в случае простого правила, такого как 

, шина вложения отношений, всегда должна возвращать меньшее правдоподобие, чем подключаемая к ней. Таким образом, для всех таких правил FSL предлагает обучать вложения отношений, избегая при этом нарушений таких неравенств. Несмотря на относительную простоту, FSL поддерживает только простые правила, в то время как KALE также поддерживает более сложные правила. Работы этих авторов представляют

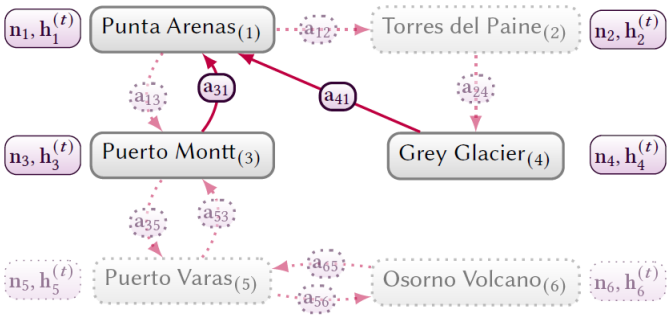
собой интересные примеры того, как дедуктивные и индуктивные формы знания – в данном случае правила и вложения – могут взаимодействовать и дополнять друг друга.

Графовые Нейронные Сети

В то время как вложения направлены на обеспечение плотного численного представления графов, пригодных для использования в существующих моделях машинного обучения, другой подход заключается в создании пользовательских моделей машинного обучения, адаптированных для данных с графовой структурой. Большинство пользовательских моделей обучения графов основаны на (искусственных) нейронных сетях, использующих естественное соответствие между ними: нейронная сеть уже соответствует взвешенному ориентированному графу, где узлы служат искусственными нейронами, а ребра -взвешенными связями (аксонами). Однако типичная топология традиционной нейронной сети – точнее, полностью связанной прямой нейронной сети – довольно однородна, будучи определенной в терминах последовательных слоев узлов, где каждый узел в одном слое связан со всеми узлами в следующем слое. И наоборот, топология графа данных довольно неоднородна, определяясь отношениями между сущностями, которые представляют его ребра. Графовая нейронная сеть (GNN) строит нейронную сеть на основе топологии графа данных, то есть узлы связаны со своими соседями в графе данных. Обычно модель затем учится сопоставлять входные объекты для узлов с выходными объектами контролируемым образом; выходные объекты, например узлы, могут быть помечены вручную или взяты из графа знаний. В отличие от вложений в графы знаний, GNN поддерживают сквозное контролируемое обучение для конкретных задач: учитывая набор помеченных примеров, GNN можно использовать для классификации элементов графа или самого графа. GNN использовались для выполнения классификации по графам, кодирующим соединения, объекты в изображениях, документах и т. д.; а также для прогнозирования трафика, построения рекомендательных систем, проверки программного обеспечения и т. д. В приведенных примерах GNN могут даже заменять алгоритмы графов; например, GNN используются для поиска центральных узлов в графах знаний контролируемым образом.

Рекурсивная графовая нейронная сеть. Рекурсивные графовые нейронные сети (Recgnn) являются основополагающим подходом к графовым нейронным сетям. Подход концептуально аналогичен систолической абстракции, показанной на рис. 25, где сообщения передаются между соседями в направлении рекурсивного вычисления некоторого результата. Однако вместо того, чтобы определять функции, используемые для определения передаваемых сообщений, мы скорее помечаем выходные данные обучающего набора узлов и позволяем фреймворку изучать функции, генерирующие ожидаемый результат, а затем применяем их для обозначения других примеров. В своей основополагающей работе Скарселли и др. предложили то, что они в общем виде называют графовой нейронной сетью (GNN), которая принимает в качестве входных данных ориентированный граф, где узлы и ребра связаны с векторами признаков, которые могут захватывать метки узлов и ребер, веса и т. д. Эти векторы признаков остаются фиксированными на протяжении всего процесса. Каждый узел в графе также связан с вектором состояния, который рекурсивно обновляется на основе информации от соседних узлов – то есть векторов признаков и состояний соседних узлов и векторов признаков ребер, простирающихся к ним/от них, – с использованием параметрической функции, называемой функцией перехода. Вторая параметрическая функция, называемая выходной функцией, используется для вычисления конечного выхода для узла на основе его собственных признаков и вектора состояния. Эти функции применяются рекурсивно до точки фиксации. Обе параметрические функции могут быть реализованы с помощью нейронных сетей, где при заданном частичном наборе контролируемых узлов в графе – то есть узлов, помеченных их желаемым выходом, – можно узнать параметры для переходных и выходных функций, которые наилучшим образом аппроксимируют контролируемые выходы. Таким образом, результат можно рассматривать как рекурсивную архитектуру нейронной

сети. Чтобы обеспечить сходимость вплоть до фиксированной точки, применяются определенные ограничения, а именно, чтобы функция перехода была исполнителем, что означает, что при каждом применении функции точки в числовом пространстве сближаются (интуитивно в этом случае числовое пространство “сжимается” при каждом применении, обеспечивая уникальную фиксированную точку).



$$\mathbf{h}_x^{(t)} := \sum_{y \in \mathrm{N}(x)} f_{\mathrm{w}}(\mathbf{n}_x, \mathbf{n}_y, \mathbf{a}_{yx}, \mathbf{h}_y^{(t-1)})$$
$$\mathbf{o}_x^{(t)} := g_{\mathrm{w}'}(\mathbf{h}_x^{(t)}, \mathbf{n}_x)$$

$$\mathbf{h}_1^{(t)} := f_{\mathrm{w}}(\mathbf{n}_1, \mathbf{n}_3, \mathbf{a}_{31}, \mathbf{h}_3^{(t-1)}) + f_{\mathrm{w}}(\mathbf{n}_1, \mathbf{n}_4, \mathbf{a}_{41}, \mathbf{h}_4^{(t-1)})$$
$$\mathbf{o}_1^{(t)} := g_{\mathrm{w}'}(\mathbf{h}_1^{(t)}, \mathbf{n}_1)$$
$$\ldots$$

На риг. 29. Слева - подграф рис. 24 выделенный район города Пунта-Аренас, где узлы помечаются векторами признаков (нx) и скрытых состояний на шаге t (ч(t) x) и ребра помечаются характеристическими векторами (аxy); справа, GNN переходов и выходов функций и пример для Пунта Аренас (x = 1), где N (x) обозначает соседние узлы x, fВт(·) обозначает переходной функции с параметрами W и gж' (·) обозначает выход функции с параметрами ж'

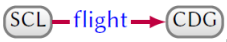
Чтобы проиллюстрировать это, рассмотрим, например, что мы хотим найти приоритетные места для создания новых туристических информационных офисов. Хорошей стратегией было бы установить их в хабах, откуда многие туристы посещают популярные направления. Таким образом, на рис.29 мы проиллюстрируем архитектуру GNN для подграфа на рис. 24, где мы выделяем окрестности of Punta Arenas этого графа, узлы которого аннотируются векторами признаков (nx) и скрытыми состояниями на шаге t (h(t) x), а ребра аннотируются векторами признаков (аx y). Векторы признаков для узлов могут, например, однократно кодировать тип узла (город, достопримечательность и т. д.), непосредственно кодировать статистику, такую как количество туристов, посещающих узел в год, и т. д. Векторы признаков для ребер могут, например, односторонне кодировать метку ребра (тип транспорта), непосредственно кодировать статистику, такую как расстояние или количество проданных билетов в год, и т. д. Скрытые состояния могут быть инициализированы случайным образом. В правой части рис. 29 представлены переходные и выходные функции GNN, где N(x) обозначают соседние узлы x, fw(·) обозначает переходную функцию с параметрами w и gw' (·) обозначает выходную функцию с параметрами w'. Пример приведен также для Пунта-Аренаса (x = 1). Эти функции будут рекурсивно применяться до тех пор, пока не будет достигнута фиксированная точка. Для обучения сети мы можем обозначить примеры мест, в которых уже есть (или должны быть) туристические офисы, и мест, в которых нет (или не должно быть) туристических офисов. Эти метки могут быть взяты из графа знаний или добавлены вручную. Затем GNN может узнать параметры w и w', которые дают ожидаемый результат для помеченных примеров, которые впоследствии могут быть использованы для маркировки других узлов. Эта модель является гибкой и может быть адаптирована по-разному: мы можем определить соседние узлы по-разному, например, относятся узлы для исходящих ребер или узлов в одном или двух переходах отсюда; мы можем позволить парам узлов быть соединенными между собой множеством ребер, мы может рассмотреть вопрос о функциях переходов и выходов с различными параметрами для каждого узла; мы можем добавить состояния и выходы для ребер; мы можем изменить сумму на другую функцию агрегирования; и т. д.

Сверточные графовые нейронные сети. Сверточные нейронные сети (CNN) получили большое внимание, в частности, для задач машинного обучения, связанных с изображениями. Основная

идея настройки изображения заключается в применении небольших ядер (так называемых фильтров) к локализованным областям изображения с помощью оператора свертки для извлечения объектов из этой локальной области. При применении ко всем локальным областям, свертка выводит карту пространственных объектов изображения. Обычно применяется несколько ядер, образующих несколько сверточных слоев. Эти ядра могут быть изучены при наличии достаточного количества маркированных примеров. Можно отметить аналогию между GNNs, как это обсуждалось ранее, и CNNs применительно к изображениям: в обоих случаях операторы применяются над локальными областями входных данных. В случае GNNs функция перехода применяется над узлом и его соседями в графе. В случае CNN свертка применяется к пикселю и его соседям на изображении. Следуя этой линии, был предложен ряд сверточных графовых нейронных сетей (ConvGNNs), где функция перехода реализуется с помощью сверток. Ключевым соображением для ConvGNN является то, как определяются области графа. В отличие от пикселей изображения, узлы в графе могут иметь различное число соседей. Это создает проблему: преимущество CNN заключается в том, что одно и то же ядро может быть применено ко всем областям изображения, но это требует более тщательного рассмотрения в случае ConvGNN, поскольку окрестности различных узлов могут быть различными. Подходы к решению этих проблем включают работу со спектральными или пространственными представлениями графов, которые индуцируют более регулярную структуру из графа. Альтернативой является использование механизма внимания для изучения узлов, особенности которых наиболее важны для текущего узла. Помимо архитектурных соображений, существует два основных различия между RecGNN и ConvGNN. Во-первых, RecGNN рекурсивно агрегируют информацию от соседей до фиксированной точки, в то время как ConvGNN обычно применяют фиксированное количество сверточных слоев. Во-вторых, RecGNN обычно используют одну и ту же функцию/параметры в однородных шагах, в то время как различные сверточные слои of a ConvGNN могут применять разные ядра/веса на каждом отдельном шаге.

Символьное Обучение

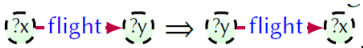
Контролируемые методы, обсуждавшиеся до сих пор, а именно вложения графов знаний и графовые нейронные сети, изучают численные модели над графами. Однако такие модели часто трудно объяснить или понять. Например, если взять граф на Рис. 30, вложения графа знаний могут предсказать, что ребро 

является очень правдоподобным, но они не дадут интерпретируемой модели, помогающей понять, почему это так: причина результата может лежать в матрице параметров, изученных для соответствия оценке правдоподобия на обучающих данных. Такие подходы также страдают от внесловарных проблемы, где они не могут обеспечить результаты для ребра с привлечением ранее недоступных вершин или ребер; например, если мы добавим узел 

, где 

это новичок в графе, графовое вложение не имеют сущности для 

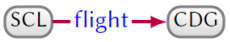
и нужно переучиваться для того, чтобы оценить правдоподобность триплета 

. Альтернативный (иногда взаимодополняющий) подход к принятию символьного обучения для того, чтобы узнать гипотезу в символьном (логическом) языке, “объясняющую” определенный набор положительных и отрицательных случаев. Эти ребра обычно генерируются из графа знаний автоматически (аналогично случаю вложений графа знаний). Гипотезы затем служат интерпретируемыми моделями, которые могут быть использованы для дальнейших дедуктивных рассуждений. Учитывая граф на рис. 30, мы можем, например, узнать правило 

, что маршруты полетов, как правило , являются маршрутами возврата. В качестве альтернативы мы могли бы изучить аксиому DL, утверждающую, что аэропорты являются либо внутренними, либо международными, либо и теми и другими: аэропорт ⊑ Внутренний аэропорт ⊔Международный аэропорт. Такие правила и аксиомы могут затем использоваться для дедуктивных рассуждений и предлагать интерпретируемую модель для нового знания, которое

влечет за собой/предсказывается; например, из вышеупомянутого правила для обратных полетов можно интерпретировать, почему предсказывается новое ребро 

. Это также дает экспертам предметной области возможность проверить модели – например, правила и аксиомы – полученные такими процессами. Наконец, правила/аксиомы количественно определены (все рейсы имеют обратный рейс, все аэропорты являются внутренними или международными и т. д.), поэтому они могут быть применены к невидимым примерам (например, с помощью вышеупомянутого правила, мы можем вывести 

новое ребро 

с невидимым узлом 

).

В этом разделе мы обсудим две формы символического обучения: интеллектуальный анализ правил, который изучает правила из графа знаний, и интеллектуальный анализ аксиом, который изучает другие формы логических аксиом.

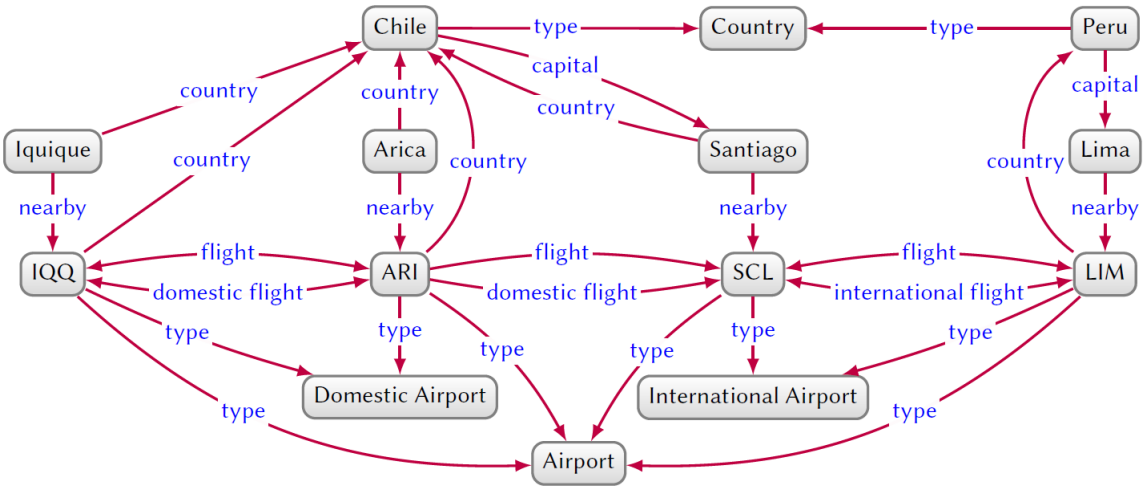
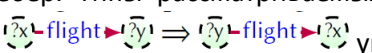
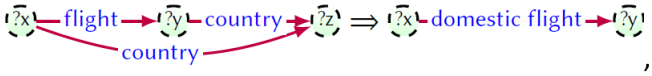


Рис. 30. Неполный направленный граф с именованными ребрами, описывающий полеты между аэропортами

Выявление правил. Выявление правил, в общем смысле, относится к обнаружению значимых паттернов в форме правил из больших коллекций фоновых знаний. В контексте графов знаний мы предполагаем, что задано множество положительных и отрицательных ребер. Обычно положительные ребра - это наблюдаемые ребра (т. е. те, которые заданы или связаны графом знаний), в то время как отрицательные ребра определяются в соответствии с заданным предположением о полноте (обсуждается позже). Цель разработки правил состоит в том, чтобы определить новые правила, которые влекут за собой высокое соотношение положительных ребер от других положительных ребер, но влекут за собой низкое соотношение отрицательных ребер от положительных ребер. Типы рассматриваемых правил могут варьироваться от более простых случаев, таких как 

упомянутые ранее, до более сложных правил, таких как 

, указание на то, что аэропорты вблизи столиц , как правило, являются международными аэропортами; или 

, указание на то, что полеты в пределах одной страны означают внутренние рейсы. В примере с международным аэропортом предполагается, что правила действуют не во всех случаях, а скорее связаны с показателями того, насколько хорошо они соответствуют положительным и отрицательным ребрам. Более подробно мы называем ребра, влекомые правилом, и множество положительных ребер (не включая само влекомое ребро) положительными влечениями этого правила. Число положительных влечений называется поддержкой правила, а отношение положительных влечений правила называется доверием к

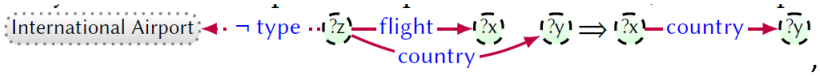
правилу. Таким образом, поддержка и доверие указывают, соответственно, на количество и соотношение “подтвержденных” последствий для правила, где цель состоит в том, чтобы определить правила, которые имеют как высокую поддержку, так и высокую уверенность. На самом деле методы интеллектуального анализа правил в реляционных установках уже давно изучаются в контексте индуктивного логического программирования (ILP). Однако графы знаний представляют собой новые проблемы из-за масштаба данных и частого предположения о неполных данных (OWA), где для решения этих проблем были предложены специальные методы. При работе с неполным графом знаний не сразу понятно, как определить отрицательные ребра. Общепринятая эвристика – также используемая для вложений графа знаний – заключается в принятии предположения о частичной полноте (PCA), которое рассматривает множество положительных ребер как те, которые содержатся в графе данных, а множество отрицательных примеров-как множество всех ребер $x\text{-}p\text{-}y'$, отсутствующих в графе, но где существует такой узел y, который $x\text{-}p\text{-}y$ находится в графе. Если взять рис. 30, то пример отрицательного ребра будет SCL–flight–ARI (при наличии SCL–flight–LIM); наоборот SCL-domestic flight-ARI, оно не является ни положительным, ни отрицательным. Тогда доверительная мера PCA - это отношение поддержки ко всем последствиям в положительном или отрицательном множестве. Например, поддержка правила ?x-domestic flight-?y ⇒ ?y-domestic flight-?x равна 2 (поскольку оно влечет IQQ-domestic flight-ARI за собой и ARI-domestic flight-IQQ на графе), в то время как уверенность равна 2/22 = 1 (отмечая , что SCL-domestic flight-ARI, хотя и влечет за собой, оно не является ни положительным, ни отрицательным и, таким образом, игнорируется мерой). Поддержка правила ?x-flight-?y ⇒ ?y-flight-?x аналогично равна 4, в то время как уверенность равна 4/55 = 0,8 (заметим, что SCL–flight–ARI это отрицательно). Таким образом, цель состоит в том, чтобы найти правила, удовлетворяющие заданным порогам поддержки и доверия. Влиятельное правило AMIE системы выявления правил для графов, которое принимает меру доверия PCA, и строит правила, сверху-вниз, начиная с головы, как правило⇒?x–country–?y. Для каждого правила голове его форме (по одному для каждой стороны метки), три вида уточнений рассчитаются, каждое из которых добавляет новое измерение в теле правила. Это новое ребро берет метку ребра из графа и может в противном случае использовать новые переменные, не появлявшиеся ранее в правиле, существующие переменные, которые уже появляются в правиле, или узлы из графа. Тогда три уточнения могут:

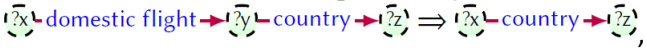
(1) Добавлено ребро с одной существующей переменной и одной новой переменной; например, уточнение вышеупомянутой головы правила может дать: ?z–flight–?x ⇒ ?x–country–?y;

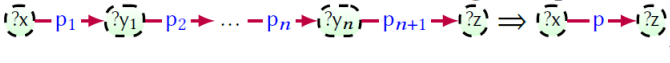
(2) Добавлено ребро с существующей переменной и узлом из графа; например, уточнение вышеуказанного правила может дать: Domestic Airport–type–?z–flight–?x ⇒ ?x–country–?y;

(3) Добавлено ребро с двумя существующими переменными; например, уточнение вышеуказанного правила может дать: Domestic Airport–type–?z–flight–?x, ?z–country–?y ⇒ ?x–country–?y.

Эти уточнения могут быть скомбинированы произвольно, что приводит к потенциально экспоненциальному пространству поиска, где поддерживаются правила, удовлетворяющие заданным порогам поддержки и доверия. Для повышения эффективности, пространство поиска может быть сокращено; например, эти три уточнения всегда уменьшают поддержку, поэтому, если правило не соответствует порогу поддержки, нет необходимости исследовать его уточнения. Дополнительные ограничения накладываются на типы генерируемых правил. Во-первых,

рассматриваются только правила до определенного фиксированного размера. Во-вторых, правила должны быть закрыты, это означает, что каждая переменная отображается по крайней мере в двух концах правила, которое гарантирует, что правила находятся в безопасности, это означает, что каждая переменная в голове появляется в хвосте; например, правила произведенные ранее, на первом и втором уточнении не закрыто (переменная y представляется один раз), ни безопасно (переменная y представляется только в голове). Чтобы обеспечить закрытые правила, третье уточнение применяется до тех пор, пока правило не будет закрыто. Для дальнейшего обсуждения возможных оптимизаций, основанных на обрезке и индексации, мы обратимся к статье о AMIE+. более поздние работы были построены на этих методах для извлечения правил из графов знаний. Gad-Elrab и соавт. предлагают метод узнать немонотонные правила – Правила сведены на нет концами в тело – для того, чтобы захватить исключения из правил, например, подход может выделить правило 

, указывающее, что рейсы в той же стране, за исключением случая, когда (отправления) аэропорт-Международный, где исключение составляет показано пунктирной и мы используем его, чтобы свести на нет преимущества. Система правил, которая также способна изучать немонотонные правила, предлагает смягчить ограничения эвристики PCA, расширив меру доверия для рассмотрения оценок правдоподобия вложений графа знаний для влекущих ребер, не появляющихся в графе. Там, где это возможно, явные утверждения о полноте графа знаний могут использоваться вместо PCA для идентификации отрицательных ребер. Таким образом, возможно использование дополнительных знаний о мощности отношений для уточнения набора отрицательных примеров и меры доверия для правил-кандидатов. В качестве альтернативы, где это возможно, онтологии могут быть использованы для получения логически определенных отрицательных ребер в рамках OWA, например, через аксиомы несвязности. Система, предлагаемая по д'Амато и соавт. использует онтологически влекомую за собой негативные ребра для определения доверие правил, сформированных на основе эволюционного алгоритма. В то время как предыдущие работы включают дискретное выделение кандидатов правил, на которых применяется скоринговая функция, другая линия исследований называют дифференциальной техникой выделения правил, которая позволяет сквозное изучение правил. Основная идея заключается в том, что соединения в телах правил могут быть представлены как матричное умножение. Более конкретно, мы можем представить отношения метки ребра p с помощью матрицы смежности Cp (размер $|V| \times |V|$), таких, что значение на i- ом ряду j - го столбца равно 1, если существует ребро помеченное p от i- й сущности к j - й сущности; в противном случае значение равно 0. Теперь мы можем представить соединение в теле правила как матричное умножение; например, учитывая 

, мы можем обозначить тело матричным умножением Adf.Ac., что дает матрицу смежности, представляющую влекущие за собой региональные узлы, где мы должны ожидать 1 в Adf.Ac... Ac.. поскольку нам даны матрицы смежности для всех меток ребер, нам остается изучить доверительные баллы для отдельных правил и изучить правила (различной длины) с пороговой доверительностью. Таким образом, NeuralLP использует механизм внимания для выбора последовательности меток ребер переменной длины для правил типа пути 

, для которых также изучаются доверительные отношения.

Выделение аксиом. Помимо правил, более общие формы аксиом, выраженные в логических языках, таких как DLs, могут быть извлечены из графа знаний. Мы можем разделить эти подходы на две категории: те, которые разрабатывают конкретные аксиомы и более общие аксиомы. Среди систем, выделяющим определенные типы аксиом, аксиомы несвязанности являются популярной целью; например, аксиома несвязанности DomesticAirport ⊓ InternationalAirport ≡ ⊥ утверждает, что пересечение двух классов эквивалентно пустому классу, или, проще говоря, ни один узел не может

быть одновременно типа Domestic Airport и International Airport. Система, предложенная Völker et al., извлекает аксиомы несвязности, основанные на (отрицательном) анализе ассоциативных правил, который находит пары классов, каждый из которых имеет много экземпляров в графе знаний, но относительно мало (или вообще нет) экземпляров обоих классов. Töpper et al. скорее извлекают несвязность для пар классов, которые имеют косинусное сходство ниже фиксированного порога. Для вычисления этого косинусного сходства векторы классов вычисляются с использованием аналогии TF–IDF, где “документ” каждого класса строится из всех его экземпляров, а “термины” этого документа являются свойствами, используемыми в экземплярах класса (сохраняя кратности). В то время как предыдущие два подхода находят ограничения несвязанности между именованными классами (например, город не связан с аэропортом), Rizzo et al. предлагают подход, который может улавливать ограничения непересекаемости между описаниями классов (например, город без ближайшего аэропорта не пересекается с городом, который является столицей страны). Подход сначала группирует похожие узлы базы знаний. Затем извлекается терминологическое дерево кластеров, где каждый конечный узел указывает на ранее извлеченный кластер, а каждый внутренний (не конечный) узел является определением класса (например, города), где левый дочерний элемент является либо кластером, имеющим все узлы в этом описание класса или подкласса (например, города без аэропортов), а правый дочерний элемент - это либо кластер, не имеющий узлов в этом классе, либо описание непересекающегося класса (например, не города с событиями). Наконец, кандидаты в аксиомы дизъюнктности предлагаются для пар описаний классов в дереве, которые, как предполагается, не имеют отношения подкласса. Другие системы предлагают методы для изучения более общих аксиом. Известной такой системой является DLLearner, которая основана на алгоритмах обучения классов (также известных как концептуальное обучение), в соответствии с которыми, учитывая набор положительных и отрицательных узлов, цель состоит в том, чтобы найти логическое описание класса, которое разделяет положительные и отрицательные множества. Например, учитывая {Iquique, Arica} как положительное множество и {Santiago} как отрицательное множество, мы можем узнать описание класса (DL) ∃поблизости. Аэропорт ⊓ ¬(∃capital−.⊤), обозначающий объекты рядом с аэропортом, которые не являются столицами, из которых все положительные узлы являются экземплярами, а никакие отрицательные узлы не являются экземплярами. Такие описания классов изучаются аналогично тому, как вышеупомянутые системы, такие как AMIE, изучают правила, с оператором уточнения, используемым для перехода от более общих классов к более конкретным классам (и наоборот), функцией оценки достоверности и стратегией поиска. Система также поддерживает изучение более общих аксиом с помощью функции подсчета очков, которая использует запросы на подсчет, чтобы определить, какое соотношение ожидаемых ребер - ребер, которые были бы влечены, если бы аксиома верна, - действительно присутствуют в графе; например, чтобы оценить аксиому «полет-.Домашний аэропорт» международный аэропорт по рисунку 30, мы можем использовать запрос графа, чтобы подсчитать, сколько узлов имеют входящие рейсы из внутреннего аэропорта (их 3), и сколько узлов имеют входящие рейсы из внутреннего аэропорта. внутренний аэропорт и международные аэропорты (их 1), где чем больше разница между обоими значениями, тем слабее доказательство аксиомы.

5. Создание и наполнение графа знаний

В этом разделе мы обсуждаем основные методы, с помощью которых можно создавать графы знаний и впоследствии обогащать их из различных источников унаследованных данных, которые могут варьироваться от простого текста до структурированных форматов (и любых других). Соответствующая методология, которой следует придерживаться при создании графа знаний, зависит от задействованных субъектов, предметной области, предполагаемых приложений, доступных источников данных и т. д. В целом, однако, гибкость графов знаний позволяет начинать

с начального ядра, которое может быть постепенно обогащенным из других источников по мере необходимости (обычно в соответствии с методологией Agile или «платой по мере использования»). В нашем текущем примере мы предполагаем, что совет по туризму решает построить граф знаний с нуля, стремясь первоначально описать основные туристические достопримечательности - места, события и т. д. - в Чили, чтобы помочь посещающим туристам определить те, которые их больше всего интересуют. Правление решает отложить добавление дополнительных данных, таких как маршруты транспорта, сообщения о преступлениях и т. д., на более поздний срок.

Человеческое Сотрудничество

Один из подходов к созданию и обогащению графов знаний - это получение прямого вклада от редакторов-людей. Таких редакторов можно найти внутри компании (например, сотрудников туристического совета), используя краудсорсинговые платформы, через механизмы обратной связи (например, туристы, добавляющие комментарии к достопримечательностям), через платформы совместного редактирования (например, вики-страницу о достопримечательностях, открытую для общедоступные редакции) и т. д. Хотя участие человека требует больших затрат, некоторые известные графики знаний были в основном основаны на непосредственном участии редакторов-людей. Однако в зависимости от того, как запрашиваются взносы, этот подход имеет ряд ключевых недостатков, в первую очередь из-за человеческой ошибки, разногласий, предвзятости, вандализма и т. д. Успешное совместное творчество также порождает проблемы, связанные с лицензированием, инструментами и культурой. Иногда люди скорее используются для проверки и кураторства дополнений к графу знаний, извлеченных другими способами (например, с помощью видеоигр с определенной целью), для определения высококачественных сопоставлений из других источников, для определения соответствующей высокоуровневой схемы и т. д.

Текстовые источники

Блоки текстов - например, из газет, книг, научных статей, социальных сетей, электронных писем, веб-сканирований и т. д. - являются богатым источником информации. Однако извлечение такой информации с высокой точностью и ее повторное использование для целей создания или обогащения графа знаний - нетривиальная задача. Для решения этой проблемы могут применяться методы обработки естественного языка (NLP) и извлечения информации (IE). Хотя процессы значительно различаются в разных средах извлечения текста, на рисунке 31 мы проиллюстрировали четыре основные задачи извлечения текста в образце предложения. Обсудим эти задачи по очереди.

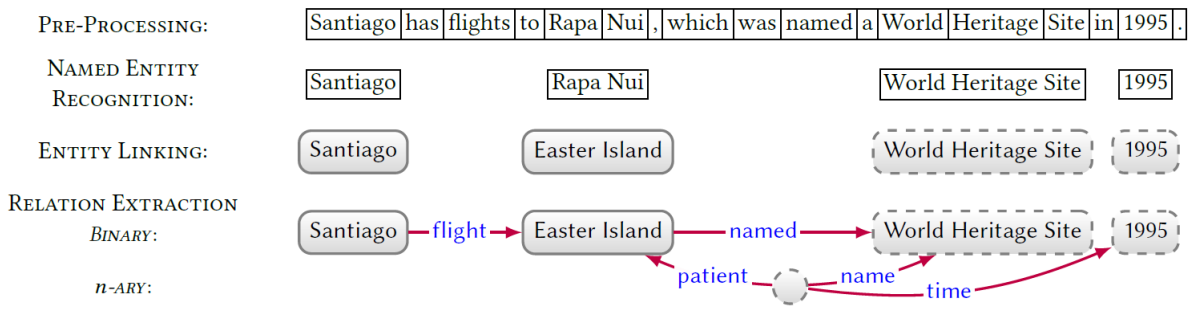


Рис. 31. Пример извлечения текста; новые узлы графа знаний показаны пунктиром

Предварительная обработка. Задача предварительной обработки может включать в себя применение различных методов к входному тексту, где на рисунке 31 показана токенизация, которая анализирует текст на элементарные термины и символы. Другие задачи предварительной обработки, применяемые к корпусу текста, могут включать: тегирование части речи (POS) для

идентификации терминов, представляющих глаголы, существительные, прилагательные и т. Д .; Анализ зависимостей, который извлекает грамматическую древовидную структуру для предложения, где листовые узлы указывают отдельные слова, которые вместе образуют фразы (например, словосочетания с существительными, словосочетания глаголов) и, в конечном итоге, предложения и предложения; и Word Sense Disambiguation (WSD) для определения значения (также известного как смысл), в котором используется слово, связывая слова с лексикой смыслов (например, WordNet или BabelNet), где, например, термин «полеты» может быть связан с WordNet воспринимает скорее «случай путешествия по воздуху», чем «лестницу между одним этажом и следующим». Подходящий тип предварительной обработки для применения часто зависит от требований более поздних задач в конвейере.

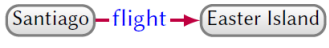
Распознавание именованных сущностей (NER). Задача NER идентифицирует упоминания именованных сущностей в тексте, обычно нацеленные на упоминания людей, организаций, местоположений и, возможно, других типов. Существует множество методов NER, со многими современными подходами, основанными на структурах обучения, которые используют лексические функции (например, теги POS, деревья синтаксического анализа зависимостей и т. д.) И географические справочники (например, списки общих имен, фамилий, стран, известных предприятий, так далее.). Контролируемые методы требуют вручную маркировать все упоминания сущностей в обучающем корпусе, тогда как подходы на основе самонастройки требуют небольшого набора начальных примеров упоминаний сущностей, из которых можно изучить шаблоны и применить их к немаркированному тексту. При дистанционном контроле известные сущности в графе знаний используются в качестве исходных примеров, с помощью которых могут быть обнаружены похожие сущности. Помимо фреймворков, основанных на обучении, правила, созданные вручную, все еще иногда используются из-за их более контролируемого и предсказуемого поведения. Именованные объекты, идентифицированные NER, могут использоваться для создания новых узлов-кандидатов для графа знаний (известных как новые объекты, показаны пунктирной линией на рисунке 31) или могут быть связаны с существующими узлами в соответствии с задачей связывания объектов, описанной ниже.

Связывание сущностей (EL). Задача EL связывает упоминания сущностей в тексте с существующими узлами целевого графа знаний, который может быть ядром создаваемого графа знаний или внешнего графа знаний. На рис. 31 мы предполагаем, что узлы Santiago и Easter Island так уже существуют в графе знаний (возможно, извлеченных из других источников). Затем EL может связать данные упоминания с этими узлами. Задача EL представляет собой две основные задачи. Во-первых, там может быть несколько способов, чтобы упомянуть одну и ту же сущность, как и в случае Rapa Nui а Easter Island; если мы создали узел Rapa Nui чтобы представить это упоминание, мы бы разделили информацию, доступную под обоими упоминаниями, по разным узлам, поэтому для целевого графа знаний важно фиксировать различные псевдонимы и многоязычные метки, с помощью которых можно ссылаться на объект. Во-вторых, одно и то же упоминание в разных контекстах может относиться к разным объектам; например, Santiago может относиться к городам Чили, Кубы, Испании и др. Таким образом, задача EL рассматривает фазу устранения неоднозначности, на которой упоминания связаны с узлами-кандидатами в графе знаний, кандидаты ранжируются и выбирается наиболее вероятный упомянутый узел. На этом этапе можно использовать контекст; например, если Easter Island есть вероятный кандидат на соответствующее упоминание рядом Santiago, мы можем повысить вероятность того, что это упоминание относится к чилийской столице, поскольку оба кандидата находятся в Чили. Другие эвристики для устранения неоднозначности рассматривают априорную вероятность, например, где Santiago чаще всего упоминается чилийская столица (будучи, например, крупнейшим городом с таким названием); для таких целей могут использоваться меры Центральности на графе знаний.

Извлечение отношения (RE). Задача RE извлекает отношения между сущностями в тексте. Самый простой случай - это извлечение бинарных отношений в закрытом окружении, в котором рассматривается фиксированный набор типов отношений. В то время как традиционные подходы часто используют шаблоны, созданные вручную, современные подходы скорее склонны использовать структуры, основанные на обучении, включая контролируемые методы, а не примеры, помеченные вручную. Другие подходы, основанные на обучении, снова используют бутстрэппинг и дистанционное наблюдение, чтобы избавиться от необходимости ручного нанесения меток; первый требует подмножества помеченных вручную исходных примеров, в то время как второй находит предложения в большом корпусе текста, в котором упоминаются пары сущностей с известным отношением / границей, которые используются для изучения шаблонов для этого отношения. Двоичный RE также может применяться с использованием неконтролируемых методов в открытой настройке - часто называемой извлечением открытой информации (OIE) - посредством чего набор целевых отношений не предопределен заранее, а извлекается из текста, например, на основе деревьев разбора зависимостей. из которых взяты отношения. Было предложено множество методов RE для извлечения-арных отношений, которые фиксируют дальнейший контекст того, как связаны сущности. На рисунке 31 мы видим, как n-арное отношение захватывает дополнительный временной контекст, обозначающий, когда Рапа Ну и был назван объектом Всемирного наследия; в этом случае анонимный узел создается для представления отношения более высокой арности в ориентированном помеченном графе. Различные методы для-арного RE основаны на семантике кадра, которая для данного глагола (например, «named») фиксирует задействованные объекты и то, как они могут быть взаимосвязаны. Ресурсы, такие как FrameNet, затем определяют фреймы для слов, например, чтобы определить, что семантический фрейм для «поименованного» включает говорящего (человека, называющего что-либо), сущность (именуемую вещь) и имя. Необязательные элементы фрейма - это объяснение, цель, место, время и т. Д., Которые могут добавить контекст отношения. Другие методы RE скорее основаны на теории представления дискурса (DRT), которая рассматривает логическое представление текста на основе экзистенциальных событий. В соответствии с этой теорией, например, наименование острова Пасхи как объекта всемирного наследия считается (экзистенциальным) событием, когда остров Пасхи является пациентом (затронутой сущностью), что приводит к логической (нео-Дэвидсоновской) формуле:

$$\exists e : \left(\text{naming}(e), \text{patient}(e, \boxed{\text{Easter Island}}), \text{name}(e, \boxed{\text{World Heritage Site}})\right)$$

Такая формула аналогична идее овеществления. Наконец, хотя отношения, извлеченные в закрытой настройке, обычно отображаются непосредственно в граф знаний, отношения, извлеченные в открытой настройке, могут нуждаться в согласовании с графом знаний; например, если процесс МЭБ извлекает бинарное отношение 

может случиться так, что у графа знаний нет других помеченных ребер, к которым есть рейсы, где выравнивание может скорее отобразить такое отношение на ребро 

предполагая, что полет используется в графе знаний. Для этих целей применялись различные методы, включая отображения и правила для выравнивания n-арных отношений; сходство, основанное на распределении и зависимости, поиск ассоциативных правил, марковская кластеризация и лингвистические методы для согласования отношений МЭБ; среди других.

Совместные задачи. Представив четыре основные задачи построения графов знаний из текста, важно отметить, что фреймворки не всегда следуют этой конкретной последовательности задач.

Рис. 32. Пример документа разметки (HTML) с исходным кодом (слева) и форматированным документом (справа)

Общей тенденцией, например, является объединение взаимозависимых задач, совместное выполнение WSD и EL, или NER и EL, или NER и RE и т. д. для взаимного улучшения производительности нескольких задач.

Размеченные текстовые источники

Сеть была основана на взаимосвязанных документах разметки, в которых маркеры (также известные как теги) используются для разделения элементов документа (обычно для целей форматирования). Большинство документов в Интернете используют язык гипертекстовой разметки (HTML). На рисунке 32 представлен пример веб-страницы в формате HTML об объектах всемирного наследия в Чили. Другие форматы разметки включают Wikitext, используемый Wikipedia, TeX для набора текста, Markdown, используемый Content Management Systems и т. Д. Один из подходов к извлечению информации из документов разметки - для создания и / или обогащения графа знаний - состоит в том, чтобы удалить маркеры (например, HTML-теги), оставляя только простой текст, к которому можно применить методы из предыдущего раздела. Однако разметка может быть полезна для целей извлечения, когда варианты вышеупомянутых задач извлечения текста были адаптированы для использования такой разметки. Мы можем разделить методы извлечения для документов разметки на три основные категории: общие подходы, которые работают независимо от разметки, используемой в конкретном формате, часто основанные на оболочках, которые сопоставляют элементы документа с выходными данными; целенаправленные подходы, нацеленные на определенные формы разметки в документе, чаще всего веб-таблицы (но иногда также списки, ссылки и т. д.); и подходы на основе форм, которые извлекают данные, лежащие в основе веб-страницы, в соответствии с понятием Deep Web. Эти подходы часто могут извлечь выгоду из закономерностей, присущих веб-страницам данного веб-сайта, будь то из-за неформальных соглашений о том, как информация публикуется на веб-страницах, или из-за повторного использования шаблонов для автоматического создания контента на веб-страницах; например, интуитивно говоря, хотя веб-страница на Рисунке 32 посвящена Чили, мы, вероятно, найдем страницы для других стран с такой же структурой на том же веб-сайте.

Извлечение на основе оберток. Многие общие подходы основаны на оболочках, которые находят и извлекают полезную информацию непосредственно из документа разметки. В то время как традиционный подход заключался в определении таких оболочек вручную - задача, для которой были определены различные декларативные языки и инструменты, - такие подходы неустойчивы к изменениям в макете веб-сайтов. Следовательно, другие подходы позволяют (полу) автоматически создавать обертки. Современный такой подход, используемый для обогащения графов знаний в

таких системах, как LODIE, заключается в применении удаленного наблюдения, при котором EL используется для идентификации и связывания сущностей на веб-странице с узлами в графе знаний таким образом, чтобы пути в разметке, соединяющие пары узлы для известных ребер могут быть извлечены, ранжированы и применены к другим примерам. Взяв, например, рисунок 32, удаленное наблюдение может связать Rapa Nui и World Heritage Sites к узлам Easter Island и World Heritage Site в графе знаний с использованием EL и с учетом ребра Easter Island –named→ World Heritage Site в графе знаний (извлеченном в соответствии с рисунком 31) определите путь-кандидат (x, td [1] - · tr− · table− · h1,) как отражающие ребра формы x –named→ y, где t [n] указывает n-й дочерний элемент тега t, t− его обратный, и 1 · t2 конкатенация. Наконец, пути с высокой степенью достоверности (например, те, которые «засвидетельствованы» многими известными ребрами в графе знаний) могут затем использоваться для извлечения новых ребер, таких как Qhapaq Ñan –named→ World Heritage Site, как на этой странице, так и на связанных страницах веб-сайта с аналогичной структурой (например, для других стран).

Report

crime	claimant	station	date
Pickpocketing	XY12SDA	Viña del Mar	2019-04-12
Assault	AB9123N	Arica	2019-04-12
Pickpocketing	XY12SDA	Rapa Nui	2019-04-12
Fraud	FI92HAS	Arica	2019-04-13

Claimant

id	name	country
XY12SDA	John Smith	U.S.
AB9123N	Joan Dubois	France
XI92HAS	Jorge Hernández	Chile

Рис. 33. пример экземпляра реляционной базы данных с двумя таблицами, описывающими данные о преступлениях

Извлечение веб-таблицы. Другие подходы нацелены на определенные типы разметки, чаще всего веб-таблицы, то есть таблицы, встроенные в веб-страницы HTML. Однако веб-таблицы предназначены для повышения удобочитаемости человеком, что часто конфликтует с машиночитаемостью. Многие веб-таблицы используются для макета и структуры страниц (например, панели навигации), тогда как те, которые содержат данные, могут соответствовать различным форматам, таким как реляционные таблицы, списки, таблицы значений атрибутов, матрицы и т. Д. Следовательно, первым шагом является классификация таблицы, чтобы найти те, которые подходят для данного механизма (-ов) извлечения. Далее, веб-таблицы могут содержать интервалы столбцов, интервалы строк, внутренние таблицы или могут быть разделены по вертикали для улучшения внешнего вида. Следовательно, фаза нормализации таблицы требуется для идентификации заголовков, слияния разделенных таблиц, отмены вложенности таблиц, транспонирования таблиц и т. Д. Впоследствии может потребоваться подход к идентификации главного героя - основного объекта, описываемого таблицей, - который скорее можно найти в другом месте в интернет страницы; например, хотя World Heritage Sites является главным героем таблицы на рисунке 31, она не упоминается в таблице. Наконец, могут применяться процессы извлечения, потенциально связывающие ячейки с объектами, столбцы с типами и пары столбцов с отношениями. В целях обогащения графов знаний более поздние подходы снова применяют дистанционное наблюдение, сначала связывая ячейки таблицы с узлами графа знаний, которые используются для генерации кандидатов для извлечения типов и отношений. Статистические распределения также могут помочь в связывании числовых столбцов. Для таблиц на определенных веб-сайтах также были разработаны специализированные структуры извлечения, где известные графы знаний, такие как DBpedia и YAGO, сосредоточены на извлечении из таблиц информационных блоков в Википедии.

Deep Web mining. Deep Web представляет собой богатый источник информации, доступный только через поиск в веб-формах, поэтому для доступа к нему требуются методы сканирования Deep Web. Были предложены системы для извлечения графов знаний из источников Deep Web. Подходы обычно пытаются генерировать разумные входные данные, которые могут быть основаны на пользовательском запросе или сгенерированы из справочных знаний, а затем извлекать данные из сгенерированных ответов (документов разметки) с использованием вышеупомянутых методов.

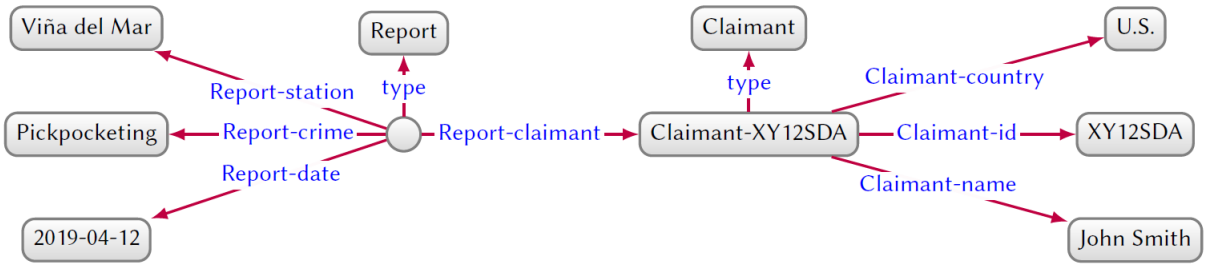


Рис. 34. Возможный результат применения прямого отображения к первым строкам обеих таблиц на рис. 33

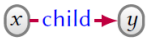
Структурированные Источники

Большая часть унаследованных данных, доступных в организациях и в Интернете, представлена в структурированных форматах, в первую очередь в таблицах - в форме реляционных баз данных, файлов CSV и т. Д. - но также и в древовидных форматах, таких как JSON, XML и т. Д. В отличие от текста и разметки документов, структурированные источники часто могут быть отображены в графы знаний, при этом структура (точно) преобразуется в соответствии с отображением, а не (неточно) извлекается. Процесс сопоставления включает в себя два этапа: 1) создание сопоставления от источника к графу и 2) использование сопоставления для материализации исходных данных в виде графа или для виртуализации источника (создание графического представления по устаревшим данным).

Отображение из таблиц. Преобладают табличные источники данных, где, например, структурированный контент, лежащий в основе многих организаций, веб-сайтов и т. Д., Размещен в реляционных базах данных. На рисунке 33 мы представляем пример экземпляра реляционной базы данных, который мы хотели бы интегрировать в наш строящийся граф знаний. Таким образом, существует два подхода к отображению содержимого из таблиц в графы знаний: прямое сопоставление и настраиваемое сопоставление. Прямое отображение автоматически генерирует граф из таблицы. На рисунке 34 мы представляем результат стандартного прямого отображения, которое создает ребро. (x)-y->(z) для каждой (не заголовочной, не пустой, не нулевой) ячейки таблицы, такой, что (x) представляет строку ячейки, имя столбца ячейки и (z) значение ячейки. В частности (x), обычно кодирует значения первичного ключа для строки (например, Claimant.id); в противном случае, если первичный ключ не определен (например, для таблицы отчета), (x) может быть анонимный узел или узел, основанный на номере строки. Метка узла (x) и ребра y дополнительно кодирует имя таблицы, чтобы избежать конфликтов между таблицами с одинаковыми именами столбцов, используемыми с разными значениями. Для каждой строки (x) мы можем добавить ребро типа на основе имени ее таблицы. Значение (z) может быть сопоставлено со значениями типа данных в соответствующей графовой модели на основе исходного домена (например, значение в столбце SQL типа Date может быть сопоставлено со значением xsd:date в модели данных RDF). Если значение равно null (или пусто), обычно соответствующее ребро будет опущено. Что касается рисунка 34, то мы выделяем различие между

узлами  и  , где первый обозначает строку (или сущность), идентифицируемую последним значением первичного ключа. В случае внешнего ключа между двумя таблицами – например, Report.Claimant .id – мы можем связать, например, с  в отличие от  , где у бывшего узла также есть имя и страна истца. Прямое сопоставление по этим направлениям было стандартизировано для сопоставления реляционных баз данных с RDF, где Stoica et al. недавно предложили аналогичное прямое отображение для графов свойств. Другое прямое сопоставление было определено для CSV и других табличных данных, которое дополнительно позволяет указывать имена столбцов, первичные / внешние ключи и типы данных, которые часто отсутствуют в таких форматах данных, как часть самого сопоставления.

Хотя прямое сопоставление может применяться автоматически к табличным источникам данных и сохранять информацию исходного источника, т.е., позволяя детерминированное обратное сопоставление, которое восстанавливает табличный источник из выходного графа, во многих случаях желательно настроить сопоставление, такие как выравнивание меток краев или узлов с графом знаний при обогащении и т. д. Наряду с этим языки декларативного сопоставления позволяют вручную определять настраиваемые сопоставления из табличных источников в графы. Стандартным языком в этом направлении является язык сопоставления RDB2RDF (R2RML), который позволяет отображать отдельные строки таблицы в одно или несколько настраиваемых ребер, при этом узлы и ребра определяются либо как константы, как значения отдельных ячеек, либо с использованием шаблонов, которые объединить несколько значений ячеек из строки и статических подстрок в один термин; например, шаблон {id} - {country} может создавать такие узлы, как  из таблицы Истца. В случае, если желаемые выходные ребра не могут быть определены из одной строки, R2RML позволяет (SQL) запросам генерировать таблицы, из которых могут быть извлечены ребра, где, например, ребра, такие как  может быть сгенерирован путем определения сопоставления по отношению к запросу, который объединяет таблицы Report и Claimant по заявителю = id, группировке по странам и применению счетчика. Затем в таблице результатов может быть определено сопоставление, так что исходный узел обозначает значение страны, граничная метка - постоянные преступления, а целевой узел - значение счетчика. Аналогичный стандарт также существует для сопоставления CSV и других табличных данных с графами RDF, снова позволяя выбирать ключи, имена столбцов и типы данных как часть сопоставления. После определения сопоставлений можно использовать их для материализации данных графа в соответствии с подходом извлечения-преобразования-загрузки (ETL), при котором табличные данные преобразуются и явно сериализуются как данные графа с использованием сопоставления. Второй вариант - использовать виртуализацию с помощью подхода Query Rewriting (QR), при котором запросы на графе (с использованием, например, SPARQL, Cypher и т. Д.) Преобразуются в запросы по табличным данным (обычно с использованием SQL). Сравнивая эти два варианта, ETL позволяет использовать данные графика, как если бы они были любыми другими данными в графе знаний. Однако ETL требует, чтобы обновления базовых табличных данных явно передавались в граф знаний, тогда как подход QR поддерживает только одну копию данных для обновления. Затем область доступа к данным на основе онтологий (OBDA) связана с подходами QR, которые поддерживают онтологические следствия, как описано в разделе 4. Хотя большинство подходов QR поддерживают только нерекурсивные следствия, выражаемые в виде одного (нерекурсивного) запроса, некоторые QR подходы поддерживают рекурсивное следование через переписывание рекурсивных запросов.

Маппинг из деревьев. Ряд популярных форматов данных основан на деревьях, включая XML и JSON. Хотя можно себе представить - не говоря уже о таких проблемах, как порядок дочерних элементов в дереве - тривиальное прямое отображение деревьев в графы путем простого создания ребер формы  для каждого узла y, который является потомком x в исходном дереве, такой

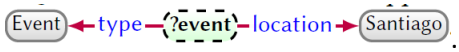
подход обычно не используется, поскольку он представляет буквальную структуру исходных данных. Вместо этого содержимое данных с древовидной структурой может быть более естественно представлено в виде графика с использованием настраиваемого сопоставления. Наряду с этим стандарт GRDLL позволяет отображать из XML в графы (RDF), а стандарт JSON-LD позволяет отображать из JSON в графы (RDF). Напротив, гибридные языки запросов, такие как XSPARQL, позволяют выполнять запросы XML и RDF интегрированным образом, таким образом поддерживая как материализацию, так и виртуализацию графов по древовидным источникам унаследованных данных.

Отображение из других графов знаний. Еще один способ построения или обогащения графов знаний - использование существующих графов знаний в качестве источника. В нашем сценарии, например, большое количество интересных мест для чилийского туристического совета может быть доступно в существующих графах знаний, таких как DBpedia, LinkedGeoData, Wikidata, YAGO, BabelNet и т. Д. Однако, в зависимости от строящегося графа знаний, не все объекты и / или отношения могут представлять интерес. Стандартный вариант для извлечения соответствующего подграфа данных - использовать запросы-конструкции SPARQL, которые генерируют графы в качестве выходных данных. Согласование сущности и схемы между графами знаний может быть дополнительно необходимо для лучшей интеграции (частей) внешних графов знаний; это может быть сделано с использованием инструментов связывания графиков на основе использования внешних идентификаторов, или действительно может быть сделано вручную. Например, Wikidata использует Freebase в качестве источника.

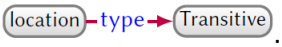
Создание Схемы/Онтологии

До сих пор обсуждение было сосредоточено на извлечении данных из внешних источников для создания и обогащения графа знаний. В этом разделе мы обсудим некоторые из основных методов создания схемы на основе внешних источников данных, включая человеческие знания. Для обсуждения извлечения схемы из самого графа знаний мы вернемся далее. В целом, большая часть работы в этой области сосредоточена на создании онтологий с использованием методологий инженерии онтологий и / или изучения онтологий. Мы обсудим эти два подхода по очереди.

Онтологическая инженерия. Инженерия онтологий относится к разработке и применению методологий для построения онтологий, предлагая принципиальные процессы, с помощью которых можно создавать и поддерживать онтологии более высокого качества с меньшими усилиями. Ранние методологии часто основывались на каскадном процессе, когда требования и концептуализация фиксировались до начала реализации онтологии на логическом языке с использованием, например, инструмента разработки онтологий. Однако для ситуаций, связанных с большими или постоянно развивающимися онтологиями, были предложены более итеративные и гибкие способы построения и поддержки онтологий. DILIGENT был ранним примером гибкой методологии, предлагая полный процесс для управления жизненным циклом онтологии и эволюции знаний, а также отделяя локальные изменения (локальные взгляды на знания) от глобальных обновлений основной части онтологии с помощью обзора процесс авторизации распространения изменений с локального на глобальный уровень. Эта методология аналогична тому, как, например, поддерживается и развивается обширная клиническая справочная терминология SNOMED CT (также доступная как онтология), где (международная) основная терминология поддерживается на основе глобальных требований, а национальные или местные расширения для SNOMED CT обслуживаются в соответствии с местными требованиями. Затем группа авторов решает, какие национальные или местные расширения распространить на основную терминологию. Более современные гибкие методологии включают в себя экстремальный дизайн (XD), модульное моделирование онтологий (MOM), упрощенную гибкую методологию разработки онтологий (SAMOD) и т. Д. Такие методологии обычно включают два ключевых элемента: требования онтологии и (в последнее время) шаблоны проектирования

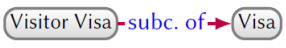
онтологий. Требования к онтологии определяют предполагаемую задачу результирующей онтологии - или даже самого графа знаний - на основе онтологии как его схемы. Обычный способ выражения требований к онтологии - это вопросы о компетенции (CQ), которые представляют собой вопросы на естественном языке, иллюстрирующие типичные знания, которые может потребоваться предоставить онтология (или граф знаний). Такие CQ могут быть затем дополнены дополнительными ограничениями и требованиями к рассуждениям в случае, если онтология также должна содержать ограничения и общие аксиомы для вывода новых знаний или проверки согласованности данных. Обычный способ тестирования онтологий (или основанных на них графов знаний) состоит в том, чтобы формализовать CQ как запросы к некоторому набору тестовых данных и убедиться, что ожидаемые результаты влекут за собой. Мы можем, например, рассмотреть CQ «Какие все события происходят в Сантьяго?», Который может быть представлен в виде графического запроса 

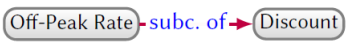
. Взяв граф данных рис. 1 и аксиомы рис. 12, мы можем проверить, связан ли ожидаемый результат 

с онтологией и данными, и поскольку это не так, мы можем рассмотреть возможность расширения аксиом, чтобы утверждать, что 

.

Шаблоны проектирования онтологий (ODP) - еще одна общая черта современных методологий, определяющая обобщенные шаблоны моделирования онтологий, которые могут использоваться в качестве вдохновения для моделирования подобных шаблонов, в качестве шаблонов моделирования или в качестве компонентов многократного использования. Несколько библиотек шаблонов были доступны в Интернете, от тщательно отобранных до открытых и модерируемых сообществом. В качестве примера, при моделировании онтологии для нашего сценария мы можем решить следовать шаблону онтологии Core Event, предложенному Krisnadhi и Hitzler, который определяет пространственно-временную протяженность, под-события и участников события, дополнительно предлагая вопросы компетентности, формальные определения и т. д., чтобы поддержать этот шаблон.

Обучение онтологий. Предыдущие методологии описывают методы, с помощью которых онтологии можно создавать и поддерживать вручную. Обучение онтологии, напротив, можно использовать для (полу) автоматического извлечения информации из текста, которая полезна для процесса разработки онтологии. Ранние методы были сосредоточены на извлечении терминологии из текста, который может представлять классы соответствующей предметной области; например, из набора текстовых документов о туризме инструмент извлечения терминологии, использующий меры единичности, которые определяют, насколько связна n-грамма как унитарная фраза, и терминология, определяющая, насколько эта фраза актуальна для предметной области, может определять n-граммы, такие как «гостевая виза», «объект всемирного наследия», «внепиковая ставка» и т. д., как терминология, имеющая особое значение для туристической сферы, и, следовательно, могут быть включены в такую онтологию. Аксиомы также могут быть извлечены из текста, где аксиомы подкласса обычно нацелены, на основе модификации существительных и прилагательных, которые постепенно специализируют концепции (например, извлечение 

от словосочетания «гостевая виза» и отдельных упоминаний слова «виза» в другом месте) или с использованием паттернов Херста (например, извлечение 

от «доступно множество скидок, например, в непиковые часы» на основе шаблона «X, например Y»). Текстовые определения также могут быть взяты из больших текстов для извлечения гиперонимных отношений и создания таксономии с нуля. Более поздние работы направлены на извлечение из текста более выразительных аксиом, включая аксиомы дизъюнктивности; и аксиомы, включающие объединение и пересечение классов, наряду с экзистенциальными, универсальными и ограниченными ограничениями мощности. Результаты процесса изучения онтологии могут затем служить входными данными для более общей

методологии инженерии онтологий, что позволяет нам проверять терминологический охват онтологии, определять новые классы и аксиомы и т.д.