УДК 81:519.768.4
DOI: 10.26140/bgz3-2021-1001-0089

СРАВНЕНИЕ СИНТАКСИЧЕСКОГО АНАЛИЗА ПРЕДЛОЖЕНИЯ ЕСТЕССТВЕННОГО ЯЗЫКА

SPIN-код: 6017-9186
AuthorID: 427788
САК Александр Николаевич, кандидат филологических наук, доцент кафедры иностранных языков и профессиональной коммуникации
SPIN-код: 5866-7742
AuthorID: 708476
БЕССОНОВА Елена Владимировна, кандидат филологических наук, доцент кафедры иностранных языков и профессиональной коммуникации
Национальный исследовательский Московский государственный строительный университет (НИУ МГСУ)
(129337, Россия, Москва, Ярославское шоссе, дом 26, e-mail: BessonovaEV@mgsu.ru)

Аннотация. При построении систем машинного перевода важной задачей является представление данных с помощью графов, где в качестве вершин выступают слова, а в качестве ребер - отношения между словами в предложении. Одной из таких задач на первом этапе анализа является классификация слов как частей речи, а на следующем этапе анализа – определение принадлежности слов к членам предложения. В статье рассматриваются методы синтаксического анализа как на основе правил, прописанных заранее с помощью традиционного объектно-ориентированного программирования, так и на основе анализа с помощью графовых сверточных нейросетей с их последующим обучением. В качестве тезауруса выступают онлайн словари.

Ключевые слова: теория графов, бинарный граф, синтаксис, семантика, графовые сверточные нейронные сети, объектно-ориентированное программирование, естественная обработка языка, функция активации, функция ошибки, матрица весов, матрица признаков, матрица степени

COMPARISON OF PARSING METHODS FOR THE ANALISYS OF A NATURAL LANGUAGE SENTENCE

SAK Alexander Nikolaevich, candidate of Philology, associate Professor of the Department of "Foreign Languages and Professional Communication"
BESSONOVA Elena Vladimirovna, candidate of Philology, associate Professor of the Department of "Foreign Languages and Professional Communication"
Moscow State University of Civil Engineering (National Research University)
(129337, Russia, Moscow, Yaroslavskoye Shosse, 26, e-mail: BessonovaEV@mgsu.ru)

Abstract. When constructing machine translation systems, an important task is to represent data using graphs, where words act as vertices, and relations between words in a sentence act as edges. One of these tasks at the first stage of the analysis is the classification of words as parts of speech, and at the next stage of the analysis to determine the belonging of words to the sentence members' classes. The article discusses methods of parsing both on the basis of rules determined in advance by means of traditional object-oriented programming, and on the basis of analysis by means of graph convolutional neural networks with their subsequent training. Online dictionaries act as a thesaurus.

Keywords: graph theory, binary graph, syntax, semantics, graph convolutional neural networks, object-oriented programming, natural language processing, activation function, error function, weight matrix, feature matrix, degree matrix.

ВВЕДЕНИЕ

Графы являются одной из важнейших областей теории вычислительных систем. Это абстрактное понятие, посредством которого можно описывать разнообразные реальные явления - организацию транспортных систем, человеческих взаимоотношений, представление структуры данных. В лингвистике теория графов также решает множество задач, связанных с представлением формальных отношений между компонентами предложения. Правильное описание предложения с помощью графа позволяет в определенной мере использовать кроме синтаксических, также и семантические связи между лексемами.

При переводе с английского языка на русский на первом этапе анализа предложения особое внимание уделяется проблеме полисемии/омонимии лексических единиц в предложении. Каждая лексема получает из онлайн словаря [1] набор частей речи, которым данное слово может соответствовать в предложении. Каждое возможное предложение, т.е. комбинацию частей речи, можно рассматривать как путь в графе, вершины которого – полный набор возможных частей речи, соответствующих данному слову [2]. Каждая возможная часть речи, соответствующая *i*-му слову соединяется ребром с каждой возможной частью речи *(i+1)* –го слова. Используя статистические данные лингвистического окружения, например, существительное реже соседствует с другими существительными, если они не выполняют функцию прилагательного, что в свою очередь должно быть отражено в словаре, можно присвоить каждой части речи, соответствующей слову, определенный вес. Самый дешевый путь по этому графу и будет наилучшая интерпретация предложения с точки зрения определения принадлежности его слов частям речи.

Определение принадлежности слов предложения к частям речи - первый шаг к его интерпретации и нахождения дерева подчинения с дальнейшим переходом к отношениям в предложении – членам предложения.

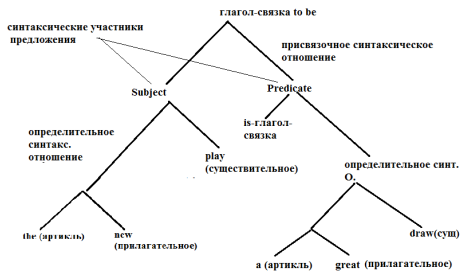

Рисунок 1 - Построение синтаксических отношений в предложении.

МЕТОДОЛОГИЯ

Для того, чтобы минимизировать ошибки при анализе предложения для построения бинарного дерева составляющих с помощью объектно-ориентированного программирования, мы руководствуемся принципом «разделяй и властвуй», а также принципом «выстре-

Baltic Humanitarian Journal. 2021. T. 10. № 1(34)
ISSN print: 2311-0066; ISSN online: 2712-9780

лил и забыл». Наш алгоритм состоит из блоков анализа разных членов предложения, и их задача - определить подчиненную часть речи и ее окружение, выделить их в отдельный связный список как элемент класса с идентифицированной вершиной, отдать ее на обработку в соответствующий блок и больше к ней не возвращаться. Чем более специализированный блок занимается обработкой соответствующего члена предложения, тем меньше вероятность ошибки и тем лучше последующие подчиненные вершины будут распознаны. С точки зрения глубокого обучения выбор графовой сверточной нейронной сети для анализа предложения обусловлен тем, что в последние годы системы, основанные на вариантах графовых нейронных сетей, таких как **(**Graph Convolutional Network), GAT (Graph Attention Network), GGNN (Gated Graph Neural Network), продемонстрировали высокую производительность при решении многих задач [3]. GCN применяют в задачах классификации текстовых данных, изображений, болезней и прогнозировании побочных эффектов. Синтаксический анализ относится как раз к проблеме классификации.

РЕЗУЛЬТАТЫ

У нас имеется класс Matrix, состоящий из строковых полей partofspeech, word, а также поля node, принадлежащему классу Wordfeatures, который включает в себя каждое слово предложения и все его употребления, взятые из онлайн словаря. Английская лексема может относиться к разным частям речи, что учитывается в онлайн словаре:

В предложении “The people gave my friend a book” имеем следующую парадигму распределения слов по частям речи:

Таблица 1 - Принадлежность слов предложения разным частям речи.

The	article		
people		noun	verb
give		noun	verb
my	pronoun		
friend	adjective	noun	verb
a	article		
book	adjective	noun	verb

Класс Matrix:

```
public class matrix
{
string partofspeech;
string word;
Wordfeatures node;
public matrix(string partofspeech, string word, Wordfeatures node)
{
this.partofspeech = partofspeech;
this.word = word;
this.node = node;
}
public string GetName()
{
return this.partofspeech;
}
public string GetWord()
{
return this.word;
}
public Wordfeatures GetNode()
{
return this.node;
} }
```

Элементы окружения определенного члена предложения включаются в связный список structure, принадлежащий классу Matrix. А уже связный список structure + вершина (определяющий член предложения) входят в другой связный список Mixture, принадлежащий к классу List

```
public class list
{
string relation;
matrix[] tArray = new matrix[10];
public list(string relation, LinkedList<matrix> sss)
{
sss.CopyTo(tArray, 0);
this.relation = relation;
}
public matrix GetPoint(int i)
{
return tArray[i];
}
public string GetRel()
{
return this.relation;
}
}
```

На первом этапе в список Mixture входит все предложение и вершина S, которая не соответствует никакому члену предложения и является формальной вершиной mixture.AddLast(new list(«S», structure));

На втором этапе анализа ищем претендента на роль глагола.

```
if (blant.Value.GetPoint(r).GetNode().GetVerb(0) != null || blant.Value.GetPoint(r).GetNode().GetPhraseVerb(0) != null)//&& blant.Value.GetPoint(r).GetNode().GetVerb(0).Length>5)
{
x = r;
if (r - 1 >= 0 && blant.Value.GetPoint(r - 1) != null)
{
if (blant.Value.GetPoint(r - 1).GetNode().GetAdverb(0) != null || blant.Value.GetPoint(r - 1).GetNode().GetNoun(0) != null || blant.Value.GetPoint(r - 1).GetName() == “pronoun”)
s++;
}
if (r + 1 <= 10 && blant.Value.GetPoint(r + 1) != null)
{
if (blant.Value.GetPoint(r + 1).GetNode().GetAdverb(0) != null || blant.Value.GetPoint(r + 1).GetNode().GetNoun(0) != null || blant.Value.GetPoint(r + 1).GetName() == “preposition” || blant.Value.GetPoint(r + 1).GetName() == “adjective” || blant.Value.GetPoint(r + 1).GetName() == “article” || blant.Value.GetPoint(r+1).GetName()==”pronoun”)
s++;
}
if (s == 2) break;
}
```

Основным признаком принадлежности к сказуемому является функция глагола или фразового глагола в одном из употреблений данной лексемы. Эта гипотеза подтверждается лингвистическим окружением глагола. В данном случае с помощью дизъюнкции проверяется, не является ли следующая за глаголом лексема предлогом, прилагательным, артиклем или местоимением. Если один из элементов соответствует условию, переменная s получает значение 1. Кстати, принцип перевода по аналогии, считается самым удачным для перевода фразовых глаголов английского языка, за счет того, что все их употребления описываются в словарных примерах, которые рефлексивны, симметричны и транзитивны соответствующим употреблениям фразовых глаголов в исходном предложении.

```
if (blant.Value.GetPoint(r - 1).GetNode().GetAdverb(0) != null || blant.Value.GetPoint(r - 1).GetNode().GetNoun(0) != null || blant.Value.GetPoint(r - 1).GetName() == «pronoun»)
```

В данном случае оператор if проверяет, является ли лексема, предшествующая предполагаемому сказуемому наречием, существительным или местоимением. Если это так, переменная s получает значение 2 и статус сказуемого у лексемы –претендента подтверждается. Так как мы знакомы с особенностями грамматики английского предложения, мы в силу нашего опыта уже

знаем все возможные элементы окружения сказуемого, что и позволяет его идентифицировать. В том случае, когда речь идет о неизвестных закономерностях - их приходится устанавливать с помощью статистических методов, речь о которых пойдет ниже.

Когда претендент на роль сказуемого подтверждается (а в английском языке это глагол или глагольная конструкция), подлежащее и его окружение засылается в коллекцию Mixture и начинается поиск подлежащего. Если подлежащее отсутствует,- возможно, это предложение в повелительном наклонении – императиве.

Двигаясь влево от сказуемого, если там имеются какие-то элементы, то заносим их в коллекцию structure, а затем в коллекцию mixture с переменной Subject для поля relation класса List.

```
for (int a = x - 1; a >= 0; a--)
{
structure.AddFirst(new matrix(blant.Value.GetPoint(a).
GetName(), blant.Value.GetPoint(a).GetWord(), blant.
Value.GetPoint(a).GetNode()));
}
if (structure.Count() > 0)
{
mixture.AddLast(new list("Subj", structure));
```

Имеем два элемента класса mixture: подлежащее со своим окружением и сказуемое со своим. Отмечаем наличие этих вершин в коллекции rlinq.

Для обработки всех возможных членов предложения, таких как подлежащее, сказуемое, обстоятельство, дополнение и т.д. имеются специальные блоки, обрабатывающие элементы класса, помечающие соответствующие вершины в бинарном графе, определяющие принадлежность оставшихся элементов другом члену предложения и делегирующие обработку более низких по статусу членов предложения с их окружением другим блокам анализа.

```
if (blant.Value.GetRel() == "Subj")
{
for (int a = 9; a >= 0; a--)
{
if (blant.Value.GetPoint(a) != null)
{
if (blant.Value.GetPoint(a).GetName() == "pronoun")
{
txt = a;
if (blant.Value.GetPoint(a).GetWord() == "I" || blant.
Value.GetPoint(a).GetWord() == "i")
sentencemembers[2, a] = "1p singular";
else
if (blant.Value.GetPoint(a).GetWord() == "you")
sentencemembers[2, a] = "2p singular";
else
if (blant.Value.GetPoint(a).GetWord() == "we")
sentencemembers[2, a] = "1p plural";
else
if (blant.Value.GetPoint(a).GetWord() == "they")
sentencemembers[2, a] = "3p plural";
else
if (blant.Value.GetPoint(a).GetWord() == "he" || blant.
Value.GetPoint(a).GetWord() == "she" || blant.Value.
GetPoint(a).GetWord() == "it")
sentencemembers[2, a] = "3p singular";
ch = 1;
for (int z = 0; z < a; z++)
if (blant.Value.GetPoint(z)!=null)
{structure.AddLast(new matrix(blant.Value.GetPoint(z).
GetName(), blant.Value.GetPoint(z).GetWord(), blant.
Value.GetPoint(z).GetNode())); }
for (int r = a + 1; r < 9; r++)
if (blant.Value.GetPoint(r)!=null)
structure1.AddLast(new matrix(blant.Value.GetPoint(r).
GetName(), blant.Value.GetPoint(r).GetWord(), blant.
Value.GetPoint(r).GetNode()));
}
}
```

В данном случае просматривается группа лексем, принадлежащая отношению Subject. Отсчет идет от последнего элемента к первому. Идет проверка, что представляет собой само подлежащее местоимение или существительное. Полученные данные заносятся как в список отношений rlinq, так и матрицу sentencemembers для описания грамматических характеристик лексемы – числа и лица.

Кроме блока анализа подлежащего имеются блок анализа сказуемого и прочих релевантных членов предложения. Надо также указать на то, что, как и при рассмотрении оптимизации арифметических выражений для их поверхностного сравнения, при построении дерева составляющих при наличии такого же члена предложения, например дополнение compliment, последующему элементу будет присвоен соответствующий индекс, для отличия его от предыдущего. Однако, несмотря на то, что данный подход основывается на достаточно глубоком знании взаимодействия членов предложения, учесть все нюансы в заранее прописанных правилах невозможно.

Тем не менее, методы анализа на основе синтеза линейной алгебры и теории графов совершенствуются и это привело к появлению нового направления в искусственном интеллекте, известном как Graph Convolutional Network[4]. Графовые сверточные сети объединяют подход к анализу на основе теории графов и сверточных нейронных сетей.

Для начала надо проанализировать данные, имеющиеся в нашем распоряжении, т.к. использование данных, например, в матрице признаков, позволит получить определенный результат даже не тренируя матрицу, что значительно сократило бы время обработки информации при реализации системы машинного перевода. Таблица 1 указывает на данные из онлайн словаря о принадлежности слов предложения разным частям речи. Можно задать соответствующие веса и включить эти веса в матрицу признаков X (feature matrix). Задаем следующие веса на входе: ***артикль- 0,1; предлог -0,2; прилагательное-0.3: наречие – 0,4; местоимение – 0,5; существительное-0,6; глагол-0,7.*** При заполнении матрицы признаков, если слово может принадлежать сразу нескольким частям речи,- эти значения суммируются. Рассмотрим уже приведенный пример предложения ***The people give my friend a book.*** Имеем следующую матрицу:

X=[0.1-0.1;1.3-1.3;1.3-1.3;0.5-0.5;1.6-1.6;0.1-0.1;1.6 -1.6];

Также нам понадобится матрица смежности. На данном этапе мы не знаем структуру предложения, поэтому зададим связи каждой вершины исключительно с соседями. Первое и последнее слово в предложении имеют формальную связь только с предыдущим и последующим словом соответственно, но мы задаем им связь с двумя предыдущими словами для выравнивания их значений.

В данном случае находим новые признаки по формуле:

$Z=ReLu(D\_hat*A\_hat*ReLu(D\_hat*A\_hat*X*W^{0})*W^{1})$ (1) [4]

```
X=[0.1-0.1;1.3 -1.3;1.3 -1.3;0.5 -0.5; 1.6 -1.6;0.1-0.1;1.6-1.6];
D=[3 0 0 0 0 0 0;0 3 0 0 0 0 0;0 0 3 0 0 0 0;0 0 0 3 0 0 0;0 0 0 0
3 0 0;0 0 0 0 0 3 0;0 0 0 0 0 0 3 ];
D_hat=inv(D);
W=[1 -1;-1 1 ];
disp (X)
disp(D_hat)
disp(A_hat)
Z=D_hat*A_hat*X*W;
Z1=D_hat*A_hat*Z*W;
disp(Z1)
```

При умножении A*X в данном виде обозначает сумму признаков их соседей. Это означает, что сверточный слой графа представляет каждый слой как совокупность его окружения. Выходит, что представление вершины

Baltic Humanitarian Journal. 2021. T. 10. № 1(34)
ISSN print: 2311-0066; ISSN online: 2712-9780

не включает его собственные признаки. Представление – это совокупность признаков соседних вершин, т.е. только вершины, имеющие петлю, включают собственные признаки в данную совокупность. Вершины с большим значением степеней будут иметь большее значение представления своих признаков, и наоборот, чем меньше степень у вершины, тем меньше значение ее признаков. Это может привести к затуханию или взрывному росту градиентов. Это также может создать проблемы при использовании алгоритмов стохастического спуска, которые используются для обучения сетей и чувствительны к диапазону значений каждого входного признака. Таким образом, надо сложить единичную матрицу I с матрицей смежности.

```
for i=1:5
A(i,i)=1;
End
```

A_hat=[1 1 1 0 0 0 0;1 1 1 0 0 0 0;0 1 1 1 0 0 0;0 0 1 1 1 0 0;0 0 0 1 1 1 0;0 0 0 0 1 1 1;0 0 0 0 1 1 1];

Представление признаков может быть нормализовано степенью вершин с помощью умножения матрицы смежности на обратную матрицу степени вершин D. Для начала найдем матрицу степеней вершин для матрицы смежности:

```
for i=1:5
s=0;
for j=1:5
s=s+A(i,j);
end
D(i,i)=s;
end
```

Таким образом, упрощенное правило распространения будет иметь вид: $f(X,A)=D^{-1}AX$

Здесь в каждом ряду значения (веса) каждого ряда матрицы смежности делятся на степень вершины, соответствующей ряду. Применяем правило распространения к трансформированной матрице смежности. В первую очередь необходимо применить веса для вершин: W=[1,-1;-1,1]; На данном этапе веса применяются произвольно.

```
D_hat=inv(D);
```

И применяем функцию активации ReLu к каждому произведению:

```
Z=D_hat*A_hat*X*W;
Z1=D_hat*A_hat*Z*W;
```

Имеем результат:

	3,7738
	3,7738
	4,0889
Z=	3,8667
	3,9556
	3,9111
	3,9111

Всего лишь на двух слоях без вычисления градиента весов W с помощью функции потерь loss=$\sum_{i=1}^{p}(ti - oi)$ получаем интересный результат: наибольший вес присваивается третьему элементу, т.е. глаголу. Элементы именных групп the people и a book получают одинаковые значения признаков соответственно, несмотря на совершенно разные значения входных элементов матрицы смежности. 4 и 5 элемент также не разошлись сильно в значениях, хотя у притяжательного местоимения значение 3,8667, а у определяющего слова - 3,9556. Очевидно, что происходит разделение предложения по группам уже на этапе предварительной подготовки двух скрытых слоев.

Мы также можем изменить формулу скрытых слоев и посмотреть, как будут меняться признаки.

$Z=softmax(A\_hat*ReLu(A\_hat*X*W^{0})W^{(1)})$ (2) [5]

В данной формуле первый слой активируется с помощью функции активации ReLu, которая активирует признак, больший нуля и не возвращает ноль в противном случае f(z)=max(0,z).

Во втором слое предполагается использование модифицированной матрицы весов W после того, как отработает функция потерь. Мы же умножаем первый слой на исходную матрицу, чтобы посмотреть результат без обучения. Второй слой активируется функцией активации softmax :

$$\sigma(z)_i=\frac{e^{Zi}}{\sum_{k=1}^{K}e^{Zk}} \quad (3)$$

```
s=0;
for i=1:n
s=s+exp(Z1(i,1)) ;
end
for i=1:n
Z1(i,1)=exp(Z1(i,1))/s;
end
```

Имеем результат:

	0,0310
	0,0310
	0,5097
Z=	0,06
	0,15
	0,1029
	0,1029

Как мы видим, сказуемое на 3-ей позиции опять получает наибольшее значение. Каждый элемент именной группы на позиции подлежащего и прямого дополнения получает одинаковые значения. Притяжательное местоимение косвенного дополнения получает наименьшее значение, а существительное косвенного дополнения имеет значение в рамках значений дополнений [0,1; 0,5].

Имеем еще одно правило многослойного распространения:

$H^{(l+1)}=\sigma(D\_hat^{-1/2}*A\_hat*D\_hat^{-1/2}*H^{(l)}W^{(l)})$ (4) [6]

где σ означает функцию активации, ReLu в данном случае, $H^{(l)} \in R^{N \times D}$ это матрица активации l-ого слоя. $H^{(0)}=X$.

Имеем следующий результат:

	3,06
	3.06
	3,2120
Z=	3,1320
	3,2040
	3,1683
	3,1683

Наблюдается схожая с предыдущими случаями картина: сказуемое опять получает наибольшее значение. Но на этот раз отличие от значения косвенного дополнения незначительно. Также наблюдаются одинаковые значения в именных группах подлежащего и прямого дополнения. Очевидно, что формула (4) наилучшим образом подходит для многослойного распространения, т.к. выявляет сказуемое со значительным отрывом от значений других элементов, четко разграничивает притяжательное местоимение и дополнение-существительное, а также указывает на отличие в значениях подлежащего и дополнения.

ВЫВОДЫ

Система синтаксического анализа предложения, основанная на правилах может показывать приемлемый результат не только на контрольном, но и на реальном примере, но при этом требует написания кода для очень большого количества случаев, которые в полной мере не могут быть охвачены даже для технического текста. В данном исследовании мы рассматриваем графовую сверточную сеть, одной из функций которой является

классификация вершин графа. Не зная синтаксическую структуру предложения, и используя только связь с соседними вершинами (словами), а также данные о принадлежности слов к определенным частям речи, мы можем достичь определенных успехов в синтаксическом анализе, даже не используя функцию потерь и без тренировки матрицы весов. В данном случае набор из трех соседних слов можно рассматривать как скользящее по тексту окно (свертку) сверточной нейронной сети, что в дополнение к заранее определяемым значениям признаков компенсирует отсутствие знаний о структуре предложения при классификации его узлов. В последующих работах мы рассмотрим построение графа синтаксических отношений в предложении на основе данного вида сетей с помощью включение в них еще одного слоя.

СПИСОК ЛИТЕРАТУРЫ:

1. Сак А.Н. Идентификация членов предложения как основная задача машинного перевода. – М.: Научное обозрение Вып.14, 2015- 9-14 сс.

2. Стивен С. Скиена. Алгоритмы. Руководство по разработке. -2-е изд.: Пер. с английского.- СПб.: БВХ-Петербург, 2011.-720 с.-66с

3. https://neerc.ifmo.ru/wiki/index.php?title=%D0%93%D1%80%D0%B0%D1%84%D0%BE%D0%B2%D1%8B%D0%B5_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8#

4. Tobias Skovgaard Jepsen. How to do Deep Learning on Graphs with Graphs Convolutional Networks https://towardsdatascience.com/how-to-do-deep-learning-on-graphs-with-graph-convolutional-networks-7d2250723780

5. Chau Pham. Graph Convolutional Networks(GCN) https://medium.com/ai-in-plain-english/graph-convolutional-networks-gcn-baf337d5cb6b

6. Thomas N. Kipf, Max Welling Semi-Supervised Classification with Graph Convolutional Networks https://arxiv.org/abs/1609.02907

7. William L. Hamilton, Rex Ying, Jure Lescovec Representation Learning on Graphs: Methods and Applications// Department of computer Science Stanford University, Stanford, CA, 94305 https://arxiv.org/abs/1709.05584

8. Jasmijn Bastings, Ivan Titov, Wilker Aziz Graph Convolutional Encoders for Syntax-aware Neural Machine translation https://www.aclweb.org/anthology/D17-1209

9. Flawson Tong Everything you need to learn about graph theory https://towardsdatascience.com/graph-theory-and-deep-learning-know-hows-6556b0e9891b

10. Trist'n Joseph The mathematics Behind Deep Learning https://towardsdatascience.com/the-mathematics-behind-deep-learning-f6c35a0fe077

11. Michael Bronstein Latent graph neural networks: Manifold learning 2.0? https://towardsdatascience.com/manifold-learning-2-99a25eeb677d

12. Connor Shorten Embedding Graphs with Deep Learning https://towardsdatascience.com/embedding-graphs-with-deep-learning-55e0c66d7752

13. Flawson Tong Graph Embedding for Deep Learning https://towardsdatascience.com/overview-of-deep-learning-on-graph-embeddings-4305c10ad4a4

14. James McCaffrey Deep Neural Networks IO Using C# https://jamesmccaffrey.wordpress.com/2017/08/02/deep-neural-network-io-using-c/

15. Hobson Lane, Cole Howard, Hannes Max Hapke Natural Language Processing in Action //Manning// Shelter Island

Статья поступила в редакцию 12.11.2020
Статья принята к публикации 27.02.2021