제가 오늘 발표하게될 논문은 2020년 VLDE학회에 발표된, streaming event detection 분야 논문이고, 논문의 title은 Data stream event prediction based on timing knowledge and state transitions입니다.

본 논문은 새로운 접근 방식을 사용하여 DATA STREAM 에서 다가오는 이벤트를 예측하는 연구에 관한 논문입니다. 이벤트 시간 순서를 이벤트 엔터티간의 관계유형으로 취급하고, 향후 이벤트 시기를 예측하기 위해서 동적 지식 그래프를 작성합니다. 기존의 지식그래프와 다르게 본 논문에서는 시간 경과에 따른 데이터 스트림의 상태를 특성화하기위해서 ‘STATES’라고 부르는 임시노드의 개념을 도입하였습니다. 관련된 이벤트 학습부터 EVENT-ORDER 그래프를 원본 데이터로부터 작성하여 예측하는 모델을 구축합니다. 결과적으로 제안된 접근방식은 기존 방식 대비 이벤트 timing 예측에 있어서 높은 예측 정확도와 recall value를 보여주었습니다. 또한, stream system이 관리할 수 있는 전체적인 throughput이 초당 1천에서 6만 튜플까지 향상되었습니다.

과거 event matching과 관련하여 많은 연구들이 진행되었지만, target event의 타이밍을 예측하는 연구는 미비하였습니다. 위 논문은 multi-attribute data stream discrete event timing prediction을 이용한 더 정확한 예측 모델을 제안합니다. 여기서 discrete이란 target event가 현재 시점에서 곧 발생할것인지, 나중에 발생할지 이분법적으로 나누었다고 생각하면 되겠습니다. 3가지의 예를 들수가 있는데 다음과 같습니다. 규칙적인 인슐린 용량, NPH 용량, 식전 또는 식후 혈당 측정, 저혈당 증상과 같은 짧은 신호를 튜플로 간주하여 실시간으로 streaming을 받음으로써 당뇨병 환자의 모니터링 및 관리에 사용하며 undesirable한 이벤트가 발생하지 않도록 방지하는 것입니다. 또한, 이와 같은 방식으로 유정 및 가스정이나 교통영역에서도 multi-attribute을 사용해서 event-timing 예측이 가능합니다.

위와 같은 real-time event prediction을 위해서는 다른 데이터 스트림 응용 프로그램 및 방법과 일관되게 예측 모델의 학습은 빠르고 실시간으로 수행되어야합니다. 또한 Traing 학습은 이상적으로 가장 최근의 소량의 데이터만을 사용하여 모델의 학습과 훈련이 데이터 스트림의 모든 변화에 적응할 수 있도록 연속적으로 진행되어야합니다. 특히 본 논문에서는 이와 관련해서 graph embedding시 새로운 접근법을 도입하고 있는데, 이는 앞서 설명한것과 같이 두개 이벤트 사이의 timing 관계를 특징으로 하는 지식그래프를 생성하고, 특정 시점에서 참으로 유지되는 이벤트집합을 특성화하는 active state 개념을 도입하였다는 점입니다. Timing 지식 그래프의 시간별 각 활성 세트와 연관된 임시노드를 임베딩시 도입하여서 target이벤트의 시점을 예측하고자 하였으며, 관련 이벤트를 학습시 tf-idf 텍스트 마이닝 기법을 사용해서 상위 k개의 관련 이벤트를 그래프의 노드로 학습하는 one-pass algorithm을 도입하였습니다.

이렇게 제안된 알고리즘의 효율성은 3가지 측면에서 검증될수가 있는데, 첫째 손실함수의 값이 빠르게 감소하고 50 epochs이후에 수렴하는 양상을 보인다는 점 입니다. 둘째, 전체적인 처리량이 1천부터 6만개의 튜플까지 handle할수있다는 점이고, 셋쨰, 예측 결과가 높은 정확도와 recall value를 보여주었다는 점입니다. 여기서 epoch는 그래프의 총 에지 수와 동일한 크기의 에지의 균일한 랜덤 샘플로 정의됩니다. 이를 통해서 훈련 알고리즘의 반복을 organize합니다. 이와 관련된 내용은 뒤에서 설명하도록 하겠습니다.

이벤트 스트림의 구성은 다음과 같습니다. 여기서 ti의 시간에 발생할 event는 ri레코드 값에 대한 Boolean predeicate입니다. T\*의 시간을 가지는 이벤트를 예로 들자면 ti에서 뺀값이 0보다크고 델타1이라는 상수보다 작으면 곧있을 이벤트로 정의하고, 여기에 델타2의 값을 더한 값이 ti보다 작으면 long after에 발생할 이벤트로 정의합니다. 자세한 내용에 앞서 background knowledge를 소개하자면 knowledge 그래프란 subject, relationship, object로 구성된 그래프입니다. 이 그래프에서의 임베딩은 각각의 노드를 다차원의 공간에 vector로 매핑하고, 잠재적인 feature를 extract하는 과정입니다. 여기서 TransE라는 distance based scoring functions 방법이 사용되었습니다. 위 방법은 training시 r을 h벡터에서부터 t벡터로의 transition으로 보고 score function을 h+t-r의 거리함수로 계산하는 기법입니다.

이벤트 타이밍 지식 그래프는 다음과 같은 state에 기반한 임베딩 기법을 통해서 작성됩니다. 기본적인 아이디어는 target event 관점에서 여러 주요 관련 이벤트를 위 그래프상 꼭짓점인 verticle V로 표현합니다. edge들은 timing 관계를 나타내고 한이벤트에서 또다른 이벤트로 향하는 엣지는 r1과 r2 총 2가지의 타입을 지닙니다. R1은 타켓이벤트 발생시점이 soon after r2는 long after를 각각 나타냅니다. 지식그래프는 특정시점에서의 state라는 개념을 가지는데 위 그래프에서 초록색 원과 빨간색 원으로 표현된것과같이 해당시점에서 ACTIVE된 곧 참인, 노드들의 집합을 나타내는 ACTIVE SET을 가집니다. 위그래프에서 Ve는 우리가 향후 예측해야하는 타겟이벤트를 나타내는데 active set과 타켓 이벤트 사이에는 Va와 같은 라이프 사이클을 가지는 임시노드 Vae라는 개념이 도입됩니다. active상태인 각 노드들로부터 임시노드까지의 edge는 이벤트 노드들 사이의 관계를 나타내고, 임시노드로부터 타겟이벤트까지의 노드는 r1과 r2에 대한 정보를 지니고있습니다.

위와 같은 그래프를 작성하기 위해서는 먼저 training data stream을 파싱하고, 그래프를 작성합니다. 이후 타겟이벤트에 대한 r1, r2관계가 정의 됩니다. 즉, target edges들이 추가됩니다. 만약 그래프 임베딩이 실시간으로 진행되며, 이후 여분의 시간이 존재할경우 non-target edges에 관해서도 real time embedding이 수행됩니다. 밑의 그림에서는 Ve’가 non-target event를 나타냅니다. 이러한 edge들은 임베딩시 낮은 우선순위를 가지게 됩니다. 밑에 보이는 Figure 3는 알고리즘의 흐름을 나타냅니다. 논문의 목적은 r1과 r2 엣지를 active set으로부터 각 이벤트까지 만드는 것인데, 이 과정에서 현시점에서 델타2길이의 시간까지 active set의 queue의 개념을 사용하게 된다. 또한 Eold라는 개념을 도입해서 이 델타2의 기간동안 한번도 발생하지 않은 이벤트들의 집합을 포함한다. 따라서, queue의 active set이 델타2의 기간이 지나면, 우리는 queue에서 그것을 제거하고 Eold에 있는 각 이벤트에대해 r2 엣지를 만들게 된다. 만약, 현 시점에서 참인 튜플이 큐에 들어가게되면, r1 엣지를 큐 내부에 있는 이벤트에 대해서 만들게 되는것입니다. 구체적인 sudo code는 다음 페이지에서 설명하도록 하겠습니다.

Event order Dynamic graph를 작성하기 위해서는 n개의 relevant이벤트와 k개의 타겟 이벤트가 주어져야합니다. 이를 이용해서 각 이벤트들의 마지막 발생 시점을 이용해서 t1부터 tn까지를 초기화합니다. 그다음 3번째 라인부터는 앞서언급한 old event를 저장할 set을 초기화하고, 지속적인 데이터의 파싱을 통해서 r1과 r2의 관계를 설정하는 것입니다.

다음은 r1엣지 관계를 설정하는 알고리즘입니다. 앞서 작성된 그래프가 input으로 들어가게되고, 각 시점을 기준으로 active set과 r1관계를 형성하고있는 이벤트 셋이 제공되고, 이를 통해서 그래프를 업데이트를 합니다. 만약 input으로 들어간 이벤트가 target이벤트일 경우 해당이벤트를 나타내는 트리플이 set of target edges에 추가됩니다. 또한 나머지 nontarget 이벤트에 대해서도 reservoir sampling을 통해서 non target r1 edge 셋에 트리플이 추가되게 됩니다.

다음은 statebased embedding과 관련된 알고리즘입니다. input으로는 dynamic event order graph stream이 들어가게되고, 임베딩 벡터와 관계가 output으로 나오게됩니다. 이 과정에서 앞서 언급한 TransE 알고리즘이 사용되게 되고, attention parameters과 함께 임시노드를 추가하게 됩니다. 1번째 줄부터 8번째 줄까지 앞서 생성된 target edge set에 존재하는 높은 우선순위를 가지는 target edges에 대해서 반복적인 훈련이 수행되고, 9번째 줄부터 모든 엣지를 대상으로 같은 작업이 반복됩니다. 이 과정에서 negative sampling이 수행되는데 큐 내부에 존재하는 positive triple에 대해서 head또는 tail의 값을 변형한 값을 batch에 포함시켜서 training을 수행한다.

다음은 target 이벤트의 예측을 위해서 relevant event를 어떻게 identify하는지를 설명하겠습니다. 이 과정에서 tf-idf라는 텍스트 마이닝 기법이 사용됩니다. 여기서 각각의 이벤트 candidates는 term에 대응되고, 타겟이벤트 전의 튜플들은 document에 대응되며, 전체 스트림에서의 일반적인 context는 text corpus에 대응됩니다. 이 과정에서 타켓 이벤트 관점에서 높은 tf-idf값을 지니는 top events를 찾기 위해서, 우리는 각 값의 집합을 basic event 와 composite 이벤트로 분할 합니다. 예를 들어서 한 개의 attribute 앞선 당뇨병 예재에서 ’식사 소화가 평소보다 느리다’라는 basic event로 분할될 수 있는데, 이와 같은 각각의 basic evnet는 general stream context에서 쉽게 발견될 수 있기 때문에 composite event 또한 사용하는것이다. 이 composite event는 term frequency관점에서 낮은 tf값을 지니지만 document 내에서 빈도가 낮으므로 idf의 값은 basic event에 비해서 높은 값을 보인다. 다만, multiple attribute가 존재하는 경우 basic event의 수는 manageable한 반면 composite수는 기하급수적으로 증가 하기 때문에 이 과정에서 bitmap operation과 a\* style pruning and bouding이라는 기법이 사용된다.

먼저 bitmap operation에 대해서 설명하자면 각 basic event에 대해서 타켓 이벤트 관점에서 발생에 대한 b1 비트맵을 생성하고, general stream context의 관점에서 발생에 대한 b2비트맵을 만들게 된다. a\* style aggressive pruning and bounding은 priority queue를 이용하는 방식입니다. 이 큐에서는 basic과 composite이벤트가 가중치와 함께 저장되고 높은 우선순위를 지니는 이벤트 부터 큐에서 pop out하게 됩니다. 이 과정에서 finalized된 이벤트의 tf-idf값은 가중치과 동일하게 되고 다른 이벤트의 upper bound보다 높은 값을 지니는 이벤트가 top event로 분류 됩니다.

다음은 이과 관련된 top relevant event를 구하는 알고리즘을 간단하게 설명하겠습니다. 첫번째 줄에서는 basic event를 할당하고 4-5 라인에서는 bitmap operation 6-29라인에서는 a\* search를 수행하게됩니다.

다음은 실험 결과입니다. 해당 실험은 위와 같은 외부 조건하에서 시행되었습니다. 첫번째 그래프는 target이벤트의 수에 따른 데이터 스트림 튜플을 초당 throughput을 나타냅니다. 이는 a\* search를 통한 top event 분류 과정에서 타겟이벤트 수가 많아지는 만큼 더 많은 데이터를 다루어야하기때문에 전체적인 throughput이 감소하는 것을 알 수 있습니다. 다음은 top relevant event의 basic event수인 cardinality에 따른 분포를 나타내는 그래프입니다. 위 그래프에서 확인할수 있듯이 cardinality의 tradeoff관계를 확인할 수 있습니다. 너무 적은 이벤트를 가지는 경우 각각의 basic 이벤트가 target event context에 나타날 확률이 높으므로 tf의 값이 높아지는 반면, 일반적인 stream context에서 자주 등장하므로 idf값이 낮아지는 것입니다.

다음 첫번째 그래프는 event order graph 생성시 퍼포먼스를 나타냅니다. 이때 relevant event수가 많을수록 퍼포먼스는 감소하고, 모든 edge들을 nontarget node에 포함시키는 것은 퍼포먼스의 감소로 이어지는 것을 알 수 있습니다. 다음은 window사이즈에 따라 training throughput을 측정한 값이다. 이때 throughput은 window사이즈가 커질수록 감소하는 경향을 나타냅니다. 다만 throughput이 높다고 해서 더 나은 예측 결과를 보장하는 것은 아닙니다. 그래프에서 알 수 있듯이 icpe가 제시된 방법중에서 가장 빠르지만 예측 정확도에서는 최악의 성능을 보여줍니다.

다음은 손실함수값의 측정치입니다. 각각의 epoch 이후에 손실 함수의 값은 급격하게 감소하는 양상을 보입니다. 다만 40에서 50 epoch에 다다랐을때에는 값이 level off하고 80에서 90 epoch에서는 0에 수렴하는 것을 알 수 있습니다.

다음은 event prediction accuracy입니다. 이 과정에서 제안된 embedding vector와 attention value의 값을 사용하였을 때 precision과 recall value를 측정하였습니다. 그래프에서 확인 할 수 있듯이, 제안된 방법은 3개의 baseline method중에서 가장 높은 정확도와 recall value를 보여줍니다. 여기서 recall value의 값이 왼쪽의 precision value보다 높은 것을 확인 할 수있는데, 이는 지식그래프의 ground truth가 완벽하지않기 때문에 real targets을 놓치는 경우에 의해서 일어납니다.

다음은 data에 따른 accuracy의 bound를 Rademacher complexity를 이용해서 측정한 결과입니다. 여기서 error bound는 샘플사이즈가 커질수록 낮아지는 양상을 보여줍니다.