Splice Junction 탐색에 특화된 기존 도구들의 분석

김소라*, 박태원*, 최석문*, 박기정**, 이도훈*, 조환규*
부산대학교 컴퓨터공학부*, 국립보건연구원 바이오과학정보과**
e-mail:{srkim_11, darkptw, csm, dohoon, hgcho}@pusan.ac.kr*,
kjpark63@gmail.com**

Analysis on Working Tools for Detecting Splice Junction

Sora Kim*, TaeWon Park*, SeokMoon Choi*,
Hwan-Gue Cho*, DoHoon Lee*
*Dept of Computer Engineering, Pusan National University

요 약

오늘날에는 HGP(Human Genome Project)로 인해 인간과 같은 고등생물은 높은 비율로, 단백질을 만들어낼 때 유전자 개수를 늘려 나가는 것이 아니라 유전자의 활용도를 높임으로써 다양한 단백질을 만들어낸다 새로운 사실이 밝혀졌다. 이로 인해 alternative splicing에 대한 관심이 높아지고 있다. Alternative splicing의 비중이 높아지며 이에 따라 이를 찾아내기 위한 다양한 방법들이 생겨나고, 이러한 방법 중 하나가 splice junction을 찾아내는 것이다. 본 논문에서는 splice junction 탐색을 위한 도구를 개발하기에 앞서 이미 기존에 존재하는 도구들을 조사하여 해당 도구들이 어떠한 사양과 알고리즘을 사용하는지를 분석 및 비교하였다.

1. 서론

오늘날, read의 길이는 점점 길어지고 있는 추세이다. 이러한 추세는 splice junction을 찾기 위함이다. 이러한 splice junction을 찾는 것은 현재 생물학 분야에서 매우 중요하게 생각하는 작업 중 하나이다. splice junction을 중요하게 생각하는 이유는 실제 유전자의 수는 그 종을 이루는 단백질의 개수보다 적기 때문이다. 이 말은 실제 하나의 유전자로부터 여러 개의 단백질이 만들어지는 복합 메커니즘이 존재한다는 사실을 알려준다.

여기서 주역이 되는 메커니즘이 바로 alternative splicing이다. Alternative splicing이란 유전자에는 여러 개의 엑손과 인트론이 존재하는데 mRNA를 만드는데 주형으로 사용되는 다양한 엑손의 다양한 조합으로부터 여러 개의 단백질이 만들어지는 메커니즘이다.

최근 미국 MIT의 Burge 팀이 최근의 high-throughput sequencing data를 사용하여 뇌, 간을 포함한 6개의 조직에서 조사한 결과 92-94%의 유전자들이 2개 이상의 mRNA를 만들어낸다는 것을 확인했다[1]. 이를 캐나다 토론토 대학의 Blencowe 팀의 연구결과와 합쳐보면, 15 종류의 조직에 있는 94%에 이르는 인간 게놈의 유전자들이 둘 이상의 mRNA를 만들어낸다[2]는 것이 밝혀진 것이다.

위의 결과를 토대로 인간과 같은 고등생물은 진화를 통해서 유전자 개수를 늘려나가는 것이 아니라 유전자의 활용도를 높이는 방법을 사용하여 더 다양한 단백질들을 만들어내는 것이다.

이렇듯 alternative splicing의 중요도가 높아짐에 따라 이를 찾고 분석하기 위한 기능 분석 기법이 요구되는 상황이다. alternative splicing을 찾기 위해 splice junction을 이용하는 방법이 있다. 따라서 splice junction 탐색을 위한 도구를 개발하기에 앞서 기존에 존재하고 있는 도구들을 분석하였다.

2. Splice Junction 탐색을 위한 도구 분석

본 논문에서는 TopHat[3], SpliceMap[4], MapSplice[5], HMMSplicer[6], SOAPsplice[7]를 분석하였다. 위의 도구들은 해당 논문에 게재된 날을 기준으로 하여 나열하였다.

아래의 표 1은 위의 5개의 도구들의 요구 사양을 비교하여 나타낸 것이다. 대부분의 도구들이 리눅스 상에서 돌아가도록 설계되었음을 확인할 수 있고, 프로그래밍 언어로 python을 선택하여 설계하였음을 확인할 수 있다. 또한 최소 메모리 요구 사양이 필요한 도구가 몇 개가 있었는데 SpliceMap의 경우 최소 8G는 있어야 하고 추천하는 바로는 16G 정도는 있어야 별 무리 없이 도구를 사용할 수 있다. SOAPsplice 같은 경우에는 6G 정도의 메모리를 보유하고 있다면 별 무리 없이 사용 가능하다.

2.1 TopHat

TopHat은 조사한 5개의 도구들 중 가장 먼저 발표된 도구이다. 해당 도구의 알고리즘은 먼저 잠정적인 엑손의 위치를 찾고 이후 이 위치에 IUM read를 맵핑시키는 방

<표 1> 기존에 존재하는 splice junction 도구들의 요구 사양.

	TopHat	SpliceMap	MapSplice	HMMSplicer	SOAPsplice
OS	Linux, MacOS X	Linux, MacOS X	64bit Linux (Ubuntu, Red Hat)	MacOS, Linux, Unix, Windows	64bit Linux
Language	C++, Python	C++	C++, Python	Python	Perl, C++(maybe)
memory	•	16G(recommed) 8G(minimum)	•	•	6G
Open Source	O	O	O	O	X

식이다.

단계별로 설명을 하면 첫 번째 단계는 Bowtie[8]를 이용하여 read들을 맵핑시킨다. 이 과정의 결과로 맵핑된 결과물로부터 잠정적인 엑손의 위치를 찾는다. 첫 번째 단계를 거치고 나면 초기에 Bowtie를 이용한 맵핑 단계에서 맵핑되지 않은 read들이 생긴다. 이러한 read들을 Initially UnMapped read라고 하여 IUM read라 명칭한다.

두 번째 단계에서는 첫 번째 단계의 결과물로 생긴 잠적인 엑손과 IUM read를 가지고 두 번째 맵핑 과정을 거치게 된다. 이 때 사용하는 방법이 seed and extend 방식이다. 이 seed and extend 방식은 첫 번째 단계에서 나온 잠정적인 엑손에서 특정 엑손 2개를 정하여 왼쪽 엑손의 끝 부분과 오른쪽 엑손의 처음 부분을 붙여 seed라 하고 이 seed를 IUM read에 맵핑시켜 맵핑되는 값이 있으면 해당 seed와 read의 나머지 부분을 점차 늘려가며 나머지 값들도 맵핑이 되는지 확인하는 방법이다.

이러한 TopHat의 방법은 몇 가지 단점을 가지고 있다. Bowtie를 이용하여 맵핑된 결과를 가지고 잠정적인 엑손의 위치를 찾아내기 때문에 만약 low sequencing depth일 경우에는 Bowtie의 threshold보다 낮은 값으로 맵핑되는 결과들이 많아지게 될 것이므로 잠정적인 엑손의 개수가 줄어들게 되고 이러한 결과는 다시 잠정적인 엑손을 이용하여 맵핑을 수행하게 되는 두 번째 단계에 영향을 주게 되므로 다양한 splice junction 부분을 찾는데 어려움이 생긴다.

또한 TopHat은 Bowtie를 사용하므로 Bowtie의 특성상 짧은 read들만을 사용하게 된다. 짧은 read를 사용하게 되면 그만큼 splice junction 부분이 나타나는 read의 양 자체가 긴 read에 비해 적기 때문에 다양한 splice junction을 찾는 데에는 어려움이 생긴다. 하지만 잠정적인 엑손을 사용하여 맵핑시키기 때문에 false negative의 값이 상대적으로 낮게 나온다는 장점이 있다.

2.2 SpliceMap

SpliceMap은 크게 4가지 단계를 가진다. 첫 번째 단계는 half-read mapping, 두 번째 단계는 seeding selection, 세 번째는 junction search, 네 번째는 paired-end filtering 단계이다. 이 때 네 번째 단계는 사용하는 read의 종류가 paired-end일 경우에만 사용되는 단계로 결과값의 정확도를 더욱 높이기 위해 사용된다.

첫 번째 단계는 read를 이등분하여 2개로 쪼개진 read를 모두 맵핑시키는 것이 아니라 2개 중에 하나만을 맵핑시킨다. 한쪽이 맵핑이 된다면 두 번째 단계인 seeding selection 단계로 맵핑된 read를 계속해서 일치하지 않는 부분이 나타날 때까지 늘려 맵핑시킨 후 맵핑된 read를 seed로 정한다. 이후 세 번째 단계인 junction search 단계로 넘어가 read의 나머지 부분을 조건에 일치하는지 확인 후 최종 맵핑시키게 되는데 이 때의 조건은 맵핑된 결과값이 seed로부터 400,000nt 안에 존재하여야 하며 만약 canonical dinucleotides splice signal이 나타난다면 이 signal을 조사하여 해당 donor와 acceptor가 일치되는지를 확인 후 이 값들이 일치된다면 seed에 맞는 나머지 read의 위치를 맵핑시키게 된다. 이러한 결과들이 모여 최종값으로 나타나게 된다.

이러한 SpliceMap 방식의 단점은 시퀀싱 에러가 있는 read를 다루기엔 효과적이지 않다는 점이다. 또한 SpliceMap은 read들이 multiple 맵핑될 경우에도 결과가 좋지 않게 나온다. 그럼에도 불구하고 SpliceMap은 read 하나를 이등분하여 그 중 하나만 맵핑시키기 때문에 다른 도구들에 비해 속도가 더 빠르다.

2.3 MapSplice

MapSplice의 알고리즘은 read를 잘게 잘라 tag라 불리는 작은 조각들로 만들고 이러한 조각들을 reference sequence에 맵핑시켜 splice junction을 찾는다. 이를 위해 MapSplice는 'tag alingment'와 'splice inference'로 크게 2단계로 나눠진다. 각 단계에는 세부 단계가 존재하는데 tag alignment의 경우 4단계, splice inference의 경우 2단계로 더 나뉘어진다.

각각의 단계에 대하여 더욱 세부적으로 말하면 첫 번째 단계는 하나의 read가 주어지면 이 read를 잘게 잘라서 tag가 불리우는 read보다 더 적은 단위의 시퀀스로 쪼개는 'partition tags into segment'이다.

두 번째 단계에서는 Bowtie 등의 도구를 이용하여 'exonic alignment'를 하는 단계로 각각의 tag들 중 exon에 완벽히 일치하는 즉, splice junction site가 나타나지

않는 tag들을 reference sequence에 맵핑시킨다. 만약 하나의 tag가 multiple alignment된다면 일단 이 단계에서는 해당되는 모든 부분에 tag를 맵핑시킨다.

세 번째는 위의 exonic alignment 시행 후 맵핑되지 못하고 남은 tag들을 'spliced alignment'하는 단계이다. 이 때 double-anchored spliced alignment를 이용하여 splice junction site가 실제 reference sequence의 어떠한 exon 사이에서 일어난 것인지 찾는다. 또한 single-anchored spliced alignment 방식을 이용하여 찾아낸다.

네 번째는 앞서 맵핑시킨 결과들을 'merging segment alignment'하는 단계이다. 만약 각각의 segment들이 유일한 지역에 맵핑되어 있고 gap없이 서로 붙어 있다면 바로 하나의 segment로 병합시키면 된다. 하지만 만약 그렇지 않고 하나의 tag가 multiple 맵핑되어 있다면 각 alignment의 조합에 대한 최적값을 계산해서 가장 좋은 값을 가지는 위치를 결과로서 가지게 된다.

다섯 번째와 여섯 번째 단계는 각각 'splice junction quality', 'best alignment for tags' 단계로 splice jucntion의 퀄리티를 계산하여 최종 값을 출력시키는 단계이다.

MapSplice의 짧은 read들을 더 짧은 tag로 만들면서 여러 군데에 맵핑될 가능성이 높아지게 되어 false netgative의 값이 높게 나온다. 이는 결과에 대한 신뢰도를 매우 떨어뜨리게 되므로 큰 단점으로 적용된다.

2.4 HMMSplicer

HMMSplicer는 Hidden Markov Model을 이용하여 머신러링을 통한 기계학습 후 이를 이용하여 read를 alignment하는 도구로 seeding reads within the reference genome, finding the splice position, matching the second piece of the read, scoring/filtering splice junction과 같은 크게 4가지 단계로 구성되고 각 단계에 세부적인 단계가 존재한다.

HMMSplicer를 돌리기 전에 pre-analysis step으로 일단 single-end read들의 FASTQ 형태만을 입력 데이터로 사용하여 Bowtie와 같은 도구들을 사용하여 전체 read sequence가 한 번에 맵핑되어 junction이 없는 read들을 걸러내는 작업을 한다. 이 작업을 통하여 splice junction이 있을 것이라 예상되는 read들만을 남기고 이렇게 남은 read들을 HMMSplicer에 사용하게 된다.

이제 전처리 단계가 끝나고 실제 HMMSplicer에 read 데이터들을 넣고 alignment 단계를 실행하게 된다. 제일 처음에는 read를 반으로 잘라 반만 맵핑시키는 것이다. 이후에 HMM을 사용하여 학습을 시킨 후 이를 이용하여 각 read-half alignment 내의 splice 위치를 밝히는 데에 사용된다. 즉, read를 반으로 자른 후 각 부분을 a와 b라고 정의한다면 splice junction site가 없는 쪽인 a를 reference sequence에 맵핑시킨 후 b를 a와 합친다. 그 후 학습된 HMM을 이용하여 splice junction 위치까지 b 부분에서 a와 같은 엑손으로부터 유래된 부분을 맵핑시킨 후 b의 남은 부분은 a가 맵핑된 엑손 이후의 엑손들 중 일치하는 곳을 찾아 맵핑시키게 된다.

이와 같은 alignment 단계가 끝난 후 scoring 단계와 filtering 단계를 거쳐 앞서 alingment 단계에서 찾았던 결과들의 신뢰도를 올린 뒤 최종 splice junction 결과를 출력하게 된다.

이러한 HMMSplicer의 단점은 현재는 다양한 방식으로 splice junction을 찾기 위해 paired-end read의 사용이 흐름을 타고 있는 데에 비해 single-end read만을 사용하여야 결과를 찾을 수 있다는 점이다. 또한 기계 학습 방법 중 하나인 HMM을 사용함으로써 한 쪽 방향으로 편중된 학습을 하게 되면 오히려 결과의 정확도가 많이 떨어지게 된다는 점이다. 이와 반대로 HMMSplicer는 기계학습을 이용하므로 종에 따른 특성을 파악하여 이에 맞게 시스템의 설정을 세팅한 후 alignment를 할 수 있으므로 다른 도구들에 비해 종의 종류에 상관없이 항상 좋은 결과를 얻을 가능성도 높다는 점이다.

2.5 SOAPsplice

SOAPsplice의 알고리즘은 그림 5와 같이 크게 세단계로 나뉜다. 첫 번째 단계에서는 다른 도구들과 마찬가지로 read를 맵핑시키는데 이 때 따로 alignment 도구를 사용하는 것이 아니라 SOAPsplice 내부에서 BWT 알고리즘을 사용하여 바로 처리하게 된다. 첫 번째 단계를 거치고 나면 IUM read들이 생기게 된다. 이러한 IUM read들은 두 번째 단계에서 IUM read들을 이등분하게 되는데 똑같은 길이로의 이등분이 아니라 다른 엑손으로부터 왔을 것이라 예상되어지는 위치에서 이등분하게 된다. 논문 상에서는 어떠한 이유를 가지고 IUM read에서 특정 위치가 다른 엑손으로부터 왔을 것이라 유추하는지에 관해서는 나와 있지 않다. 이렇게 read를 이등분한 후에는 아래의 5가지의 조건을 만족하는 위치를 찾게 된다.

- 이등분된 각각의 segment들은 threshold 값보다 커야 되는데 이 threshold 값의 디폴트 값은 8bp이다.
- 각 segment들을 맵핑시킬 때 한 개의 미스매치를 허용하고 갭은 허용하지 않는다.
- 두 segment의 사이는 인트론이 있을 것이라 예상하는 거리는 50bp에서 50,000bp 사이라 생각한다. 이 거리는 진핵생물의 알려진 인트론 사이즈의 대다수를 차지하는 값이다.
- 인트론의 가장자리는 "GT-AG", "GC-AG", "AT-AC"와 같은 형태여야 한다. 만약 이등분된 IUM read들이 multiple hit이 될 경우에 "GT-AG" 가장자리가 가장 높은 우선권을 가지고 "GC-AG", "AT-AC" 순으로 splice junction 후보가 될 곳을 찾게 된다.
- segment들이 multiple hit이 될 경우, 오직 한 segment는 유일하게 맵핑되지만 나머지 한 쪽이 multiple hit일 경우나 각 segment들이 많아봐야 3군데의 hit이 고려될 때만을 다룬다.

<표 2> 기존에 존재하는 도구들의 탐색 방법 비교.

		TopHat	SpliceMap	MapSplice	HMMSplicer	SOAPsplice
pre-processing tool		Bowtie	Bowtie	Bowtie	Bowtie	using BWT
Main method		seed and extend alignment	half-read mapping	tag alignment	HMM & half-read mapping	two segments, derived from different exons
Input file	read	FASTA/FASTQ, single/paired-end	FASTQ, single/paired-end	FASTA/FASTQ, single/paired-end	FASTQ, single	FASTA/FASTQ, single/paired-end
	reference	FASTA	FASTA	FASTA	FASTA	FASTA
Output file		BEM, BED	SAM, BED, WIG	SAM, BED	BED	JUNC

위의 5가지 조건을 모두 만족하게 되는 위치를 찾아 IUM read를 맵핑시키게 된다. 이 때 만약 read의 길이가 50bp를 넘을 경우에는 만약 read의 길이가 100bp보다 짧다면 read를 똑같은 길이로 이등분하고 100bp보다 짧지 않다면 50bp의 값을 가지도록 multiple segment를 만들고 이를 sub-read라 칭한다. 이후 이 sub-read들을 가지고 첫 번째 단계부터 똑같은 과정을 거치게 된다.

이러한 SOAPsplice 방식의 단점은 짧은 read일 때는 계산 시간이 크게 오래 걸리지 않고 다른 도구들과 비슷한 시간대인 것으로 나타나지만 read가 긴 경우에는 다른 도구들보다 적게는 2배에서 많게는 9배 정도 차이가 나는 것으로 조사되었다. 하지만 SOAPsplice의 경우 매우 늦은 expression level에서도 splice junction을 찾는 데에 어려움이 없는 것이 장점이다.

3. 결론

표 2는 본 논문에서 분석한 다양한 도구들의 splice junction 탐색 방법 데이터를 정리한 것이다. 표의 pre-processing tool은 5개의 도구가 실제 splice junction을 탐색하기 위한 알고리즘을 사용하기 전에 데이터를 splice junction이 있는 read들로 걸러내기 위해 사용하는 도구를 뜻한다. SOAPsplice는 따로 기존에 존재하는 도구를 사용하지 않고 내부적으로 BWT 알고리즘을 이용하여 처리한다. 각 도구들은 대용량 파일 처리를 위해 모두 리눅스 상에서 실행 가능하도록 만들어져 있었고 대부분 실제 alignment 부분은 C와 python을 많이 사용하였고, 스크립트 등은 perl과 python을 이용하여 만들어져 있었다.

또한 대부분의 도구들이 크게 2가지 단계로 정의되어 있는데 첫 번째 단계는 pre-processing alignment, 두 번째 단계는 junction alignment와 같은 형태로 되어 있다. 처음 단계에서 수많은 read들 중 IUM read를 걸러낸 후 이 IUM read가 splice junction site를 가지고 있다는 가정하에 IUM read를 다양한 형태로 분할한 후 분할된 segment들을 다시 맵핑시키므로써 splice junction site를 찾게 되는 것이다.

따라서 앞으로 설계할 splice juction을 탐색하는 도구에는 이러한 두 단계를 그대로 사용하고 다르게 가야 할 부분은 IUM read를 처리할 새로운 방법을 설계하여 false negative가 크지 않고 속도를 빠르게 하는 알고리즘을 설계할 예정이다.

Acknowledge

본 연구는 질병관리본부 학술연구용역사업(2011-E72002-00) 으로 수행 되었습니다.

참고문헌

[1] E. T. Wang and R. Sandberg et al., “Alternative isoform regulation in human tissue transcriptomes,” Nature, Vol. 456, No. 27, pp 470-476, 2008.

[2] Q. Pan and O. Shai et al., “Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing,” Nature Genetics, Vol. 40, No. 12, pp 1413-1415, 2008.

[3] C. Trapnell, L. Pachter and S. L. Salzberg, “TopHat: discovering splice junctions with RNA-Seq,” Bioinformatics, Vol. 25, No. 9, pp. 1105-1111, 2009.

[4] K. F. Au and H. Jiang et al., “Detection of splice junctions from paired-end RNA-seq data by SpliceMap,” Nucleic Acids Research, Vol. 38, No. 14, pp. 4570-4578, 2010.

[5] K. Wang and D. Singh et al., “MapSplice: Accurate mapping of RNA-seq reads for splice junction discovery,” Nucleic Acids Research, Vol. 38, No. 18, pp. e178, 2010.

[6] M. T. Dimon, K. Sorber and J. L. DeRisi, “HMMSplicer: A Tool for Efficient and Sensitie Discovery of Known and Novel Splice Junctions in RNA-Seq Data,” PLoS One, Vol. 5, No. 11, pp. e1375, 2010.

[7] S. Huang and J. Zhang et al., “SOAPsplice: genome-wide *ab initio* detection of splice junctions from RNA-Seq data,” frontiers in GENETICS, Vol. 2, No. 0, doi:10.3389/fgene.2011.00046, 2011.

[8] B. Langmead, C. Trapnell, M. Pop and S. Salzberg. “Ultrafast and memory-efficient alignment of short DNA sequences to the human genome,” Geonme Biology, Vol. 10, pp. R25, 2009

[9] T. W. Lam and W. K. Sung, et al., “Compressed indexing and local alignment of DNA,” Bioinformatics, Vol. 24, No. 6, pp. 791-797