

Dependency Parsing

Lecture 2

Overview

- Nivre's Arc-Eager / Arc-Standard Algorithm
- Covington's Parsing Strategy
- Pseudo-projective Parsing
- Java ML Libraries
 - OpenNLP MaxEnt
 - Mallet
 - Liblinear

Arc-Standard Algorithm

- Two stacks:
 - Stack with tokens that still have to be processed (**Input Stack**)
 - Stack with tokens that already have been processed (**Processed Stack**)
- One set:
 - Set of predicted dependency pairs
- Initial configuration:
 - **Processed Stack** with “0” on top of it. **Input Stack** with the whole input on it.
- Terminal configuration:
 - Empty **Processed Stack**

Arc-Standard Parsing Algorithm

- **Leftarc**
 - Makes the top token of the **input stack** the head of the top token of the **processed stack**
 - Pops the top token from the **processed stack**
- **Rightarc**

*if the top token of the **processed stack** already has been attached to all of its dependents to the right :*

 - Makes it the head of the top token of the **input stack**
 - Pops the top token from the **input stack**
 - Moves the top token of the **processed stack** on top of the **input stack**
- **Shift**
 - Moves the top token of the **input stack** on top of the **processed stack**

Input

1	A	DT	—	—
2	hearing	NN	—	—
3	on	IN	—	—
4	the	DT	—	—
5	issue	NN	—	—
6	is	VBZ	—	—
7	scheduled	VBN	—	—
8	today	NN	—	—
9	.	.	—	—

Arc-Standard Parsing Example

[]

[0]

[9, 8, 7, 6, 5, 4, 3, 2, 1]

Shift

0 A₁ hearing₂ on₃ the₄ issue₅ is₆ scheduled₇ today₈ .₉

Arc-Standard Parsing Example

[]

[0, 1]

[9, 8, 7, 6, 5, 4, 3, 2]

Leftarc

0 A1 hearing2 on3 the4 issue5 is6 scheduled7 today8 .9

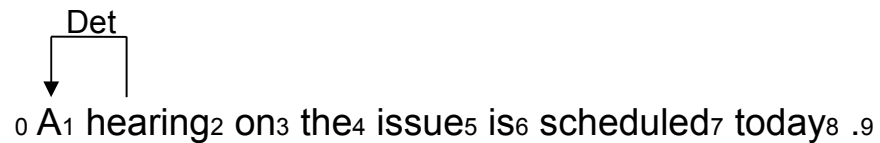
Arc-Standard Parsing Example

[(1,2)]

[0]

[9, 8, 7, 6, 5, 4, 3, 2]

Shift



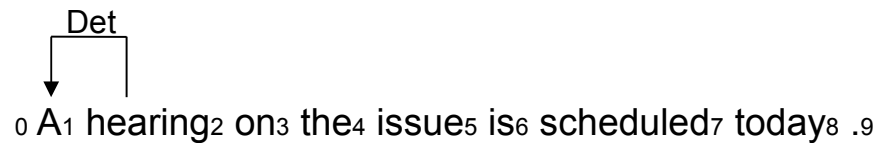
Arc-Standard Parsing Example

[(1,2)]

[0, 2]

[9, 8, 7, 6, 5, 4, 3]

Shift



Arc-Standard Parsing Example


[(1,2)]

[0, 2, 3]

[9, 8, 7, 6, 5, 4]

Shift

0 A1 hearing2 on3 the4 issue5 is6 scheduled7 today8 .9



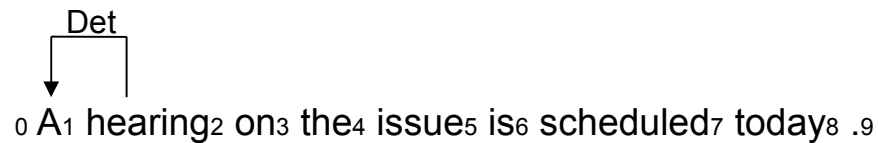
Arc-Standard Parsing Example

[(1,2)]

[0, 2, 3, 4]

[9, 8, 7, 6, 5]

Leftarc



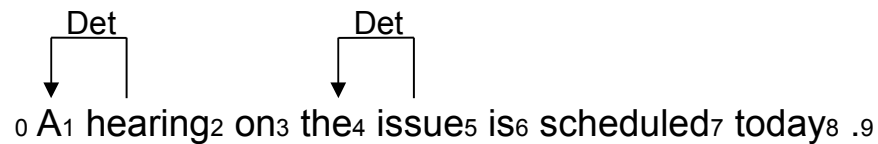
Arc-Standard Parsing Example

[(1,2), (4,5)]

[0, 2, 3]

[9, 8, 7, 6, 5]

Rightarc



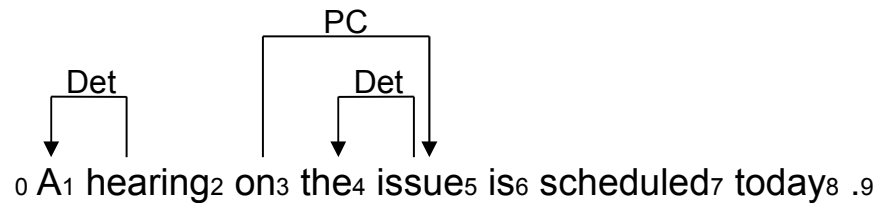
Arc-Standard Parsing Example

[(1,2), (4,5), (5,3)]

[0, 2]

[9, 8, 7, 6, 3]

Rightarc



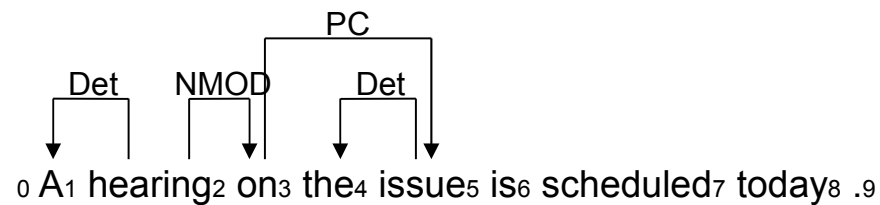
Arc-Standard Parsing Example

[(1,2), (4,5), (5,3), (3,2)]

[0]

[9, 8, 7, 6, 2]

Shift



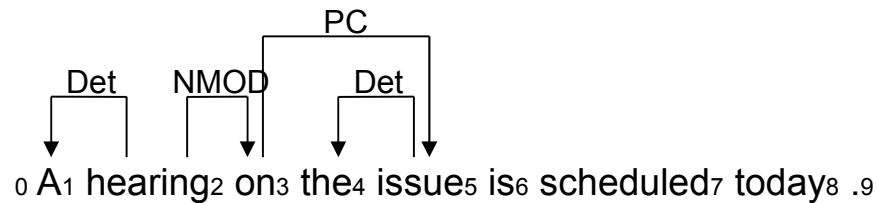
Arc-Standard Parsing Example

[(1,2), (4,5), (5,3), (3,2)]

[0, 2]

[9, 8, 7, 6]

Leftarc



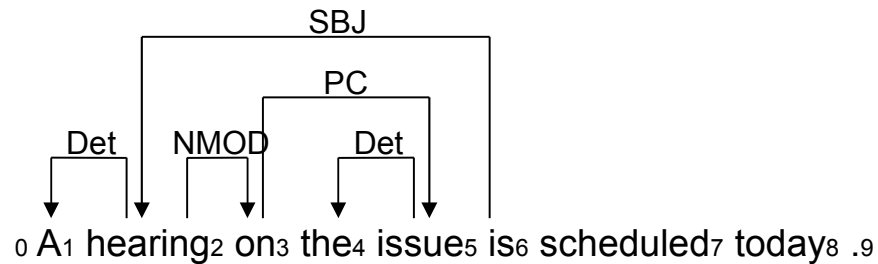
Arc-Standard Parsing Example

[(1,2), (4,5), (5,3), (3,2), (2,6)]

[0]

[9, 8, 7, 6]

Leftarc



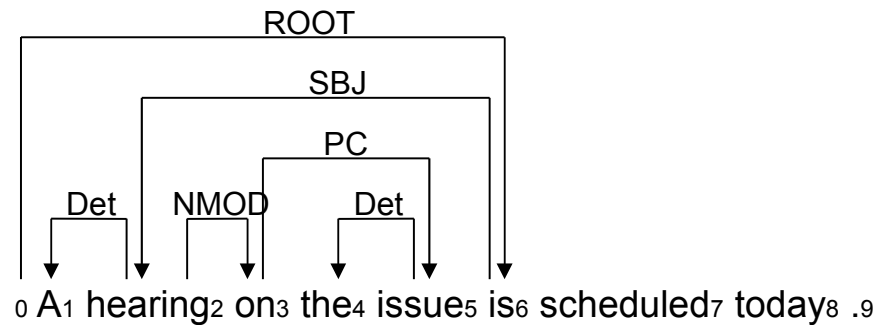
Arc-Standard Parsing Example

[(1,2), (4,5), (5,3), (3,2), (2,6), (6,0)]

[]

[9, 8, 7, 6]

Shift



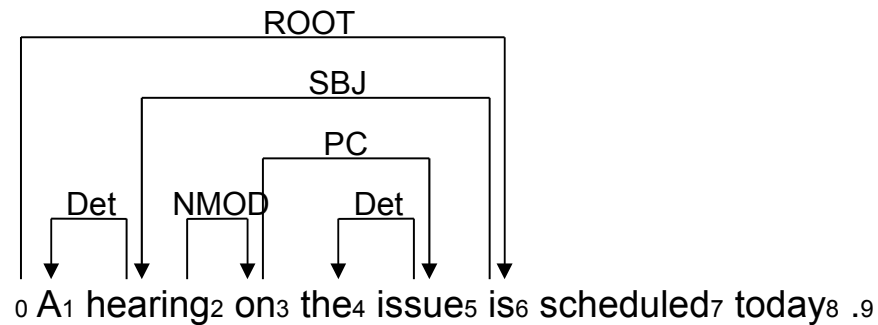
Arc-Standard Parsing Example

[(1,2), (4,5), (5,3), (3,2), (2,6), (6,0)]

[6]

[9, 8, 7]

Shift



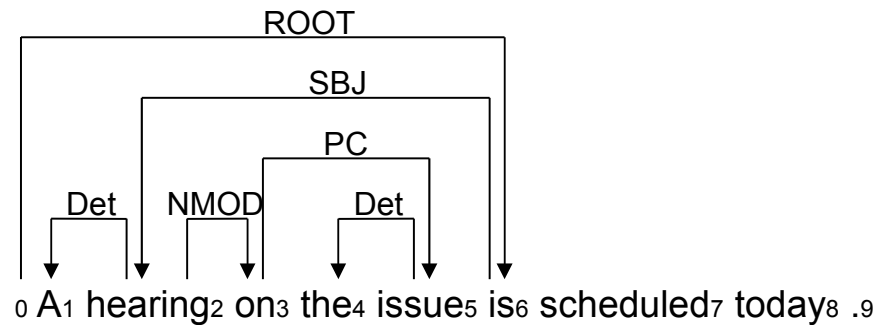
Arc-Standard Parsing Example

[(1,2), (4,5), (5,3), (3,2), (2,6), (6,0)]

[6, 7]

[9, 8]

Rightarc



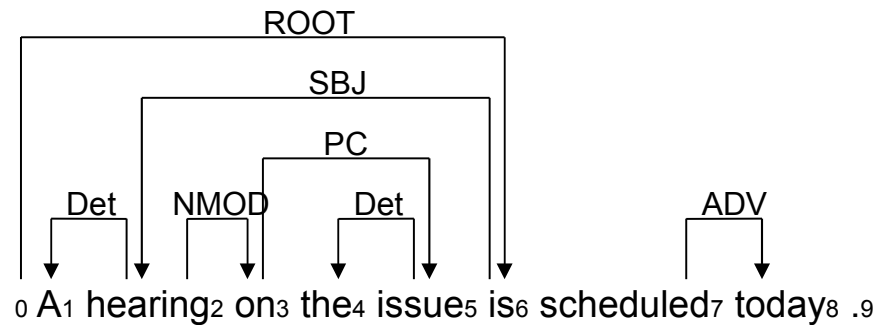
Arc-Standard Parsing Example

[(1,2), (4,5), (5,3), (3,2), (2,6), (6,0), (8,7)]

[6]

[9, 7]

Rightarc



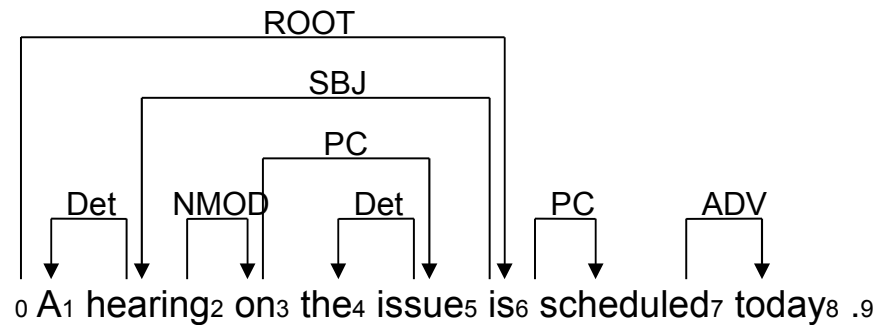
Arc-Standard Parsing Example

[(1,2), (4,5), (5,3), (3,2), (2,6), (6,0), (8,7), (7,6)]

[]

[9, 6]

Shift



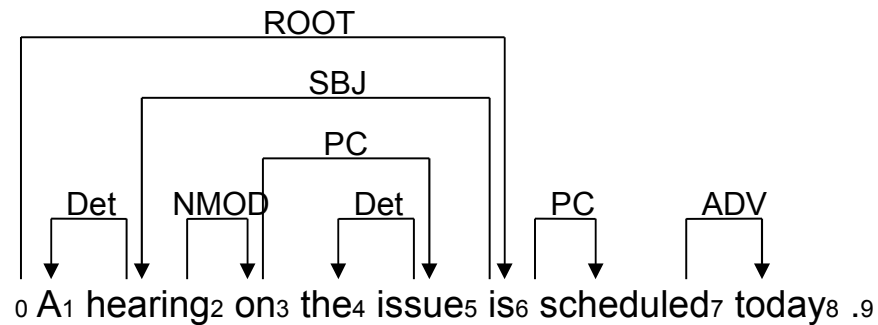
Arc-Standard Parsing Example

[(1,2), (4,5), (5,3), (3,2), (2,6), (6,0), (8,7), (7,6)]

[6]

[9]

Rightarc



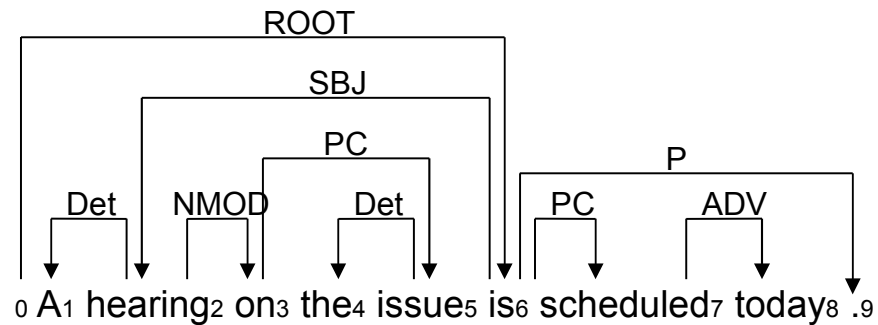
Arc-Standard Parsing Example

[(1,2), (4,5), (5,3), (3,2), (2,6), (6,0), (8,7), (7,6), (9,6)]

[]

[6]

Shift



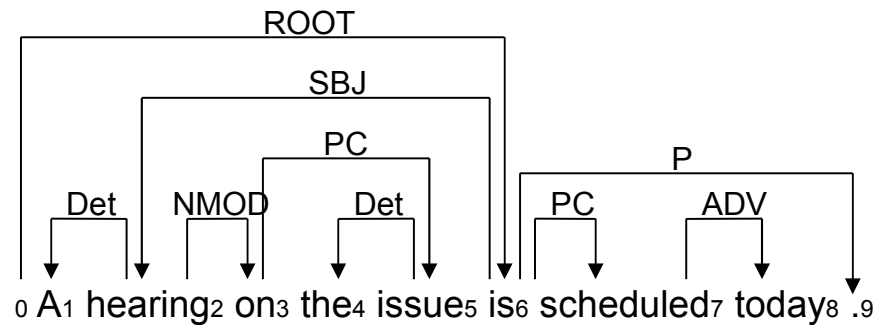
Arc-Standard Parsing Example

[(1,2), (4,5), (5,3), (3,2), (2,6), (6,0), (8,7), (7,6), (9,6)]

[6]

[]

Shift



Output

1	A	DT	2	DET
2	hearing	NN	6	SBJ
3	on	IN	2	NMOD
4	the	DT	5	DET
5	issue	NN	3	OC
6	is	VBZ	0	ROOT
7	scheduled	VBN	6	PC
8	today	NN	7	ADV
9	.	.	6	P

Fundamental Parsing Algorithm

- Go through all words in the sentence and for every word j try to combine it with the words $1..j-1$
- Dependents of any given word are more likely to be near it than far away, therefore work backwards through $j-1, j-2$ etc, rather than forwards from 1 to $2, 3$ etc.
- Don't consider not permissible pairs (don't look for a parent if a word already has one, pairs which introduce cycles etc.)

Fundamental Parsing Algorithm

Operations

- Given the current word j and a word $i < j$ there are three operations:
 - **Link(i, j)**
 - Creates a dependency pair with j being the head of i
 - **Link(j, i)**
 - Creates a dependency pair with i being the head of j
 - **Shift**
 - Rejects the current pair and moves on

Input

1	She	PRP	—	—
2	was	VBD	—	—
3	the	DT	—	—
4	first	JJ	—	—
5	woman	NN	—	—
6	to	TO	—	—
7	be	VB	—	—
8	appointed	VBN	—	—
9	FCC	NNP	—	—
10	general	JJ	—	—
11	counsel	NN	—	—
12	.	.	—	—

Fundamental Parsing Algorithm Example

[]

j: 1

i: 0

{l1=0.016, **shift**=0.9832}

0 She₁ was₂ the₃ first₄ woman₅ to₆ be₇ appointed₈ FCC₉ general₁₀ counsel₁₁ .₁₂

Fundamental Parsing Algorithm Example

[]

j: 2

i: 1

{I2=0.8848, I1=0.0083, shift=0.1069}

0 She₁ was₂ the₃ first₄ woman₅ to₆ be₇ appointed₈ FCC₉ general₁₀ counsel₁₁ .₁₂

Fundamental Parsing Algorithm

Example

[(1,2)]

j: 2

i: 0

{I1=0.6437, shift=0.2547}

0 She₁ was₂ the₃ first₄ woman₅ to₆ be₇ appointed₈ FCC₉ general₁₀ counsel₁₁ .₁₂

```
graph TD
    SBJ[SBJ] --> She[She]
    SBJ --> P[was the first woman]
```

Fundamental Parsing Algorithm

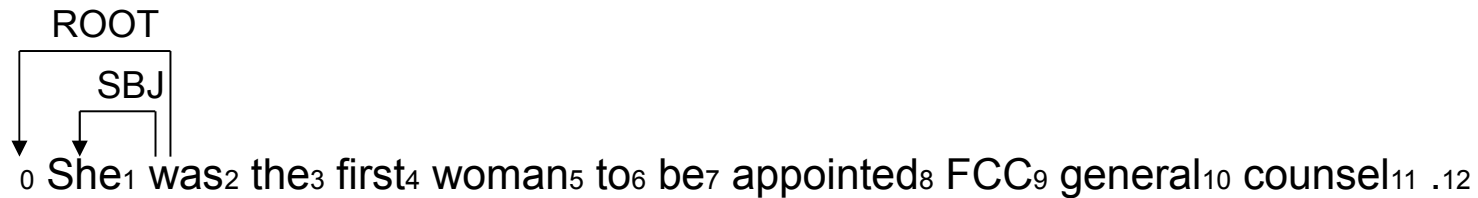
Example

[(1,2), (2,0)]

j: 3

i: 2

{I1=0.6437, shift=0.2547}



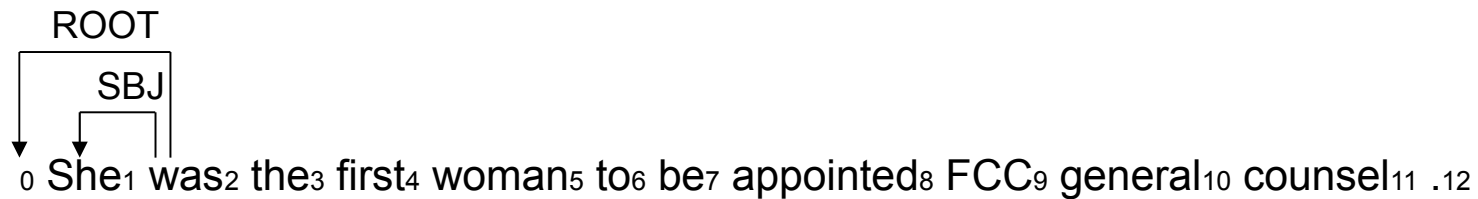
Fundamental Parsing Algorithm Example

[(1,2), (2,0)]

j: 4

i: 3

{l2=0.0224, l1=0.0, **shift**=0.9776}



Fundamental Parsing Algorithm

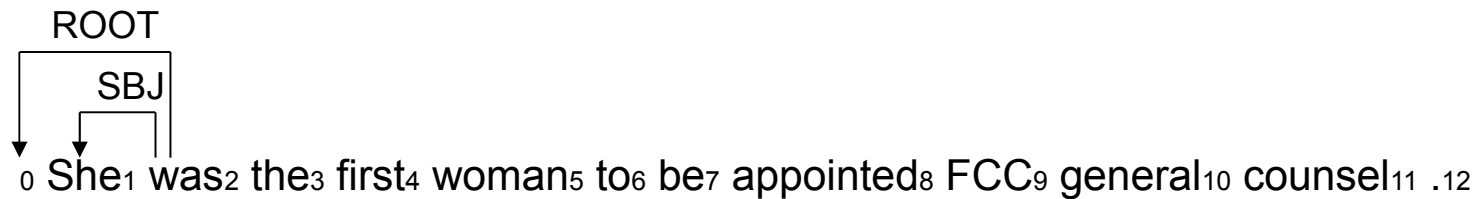
Example

[(1,2), (2,0)]

j: 4

i: 2

{l1=0.0745, **shift**=0.9247}



Fundamental Parsing Algorithm

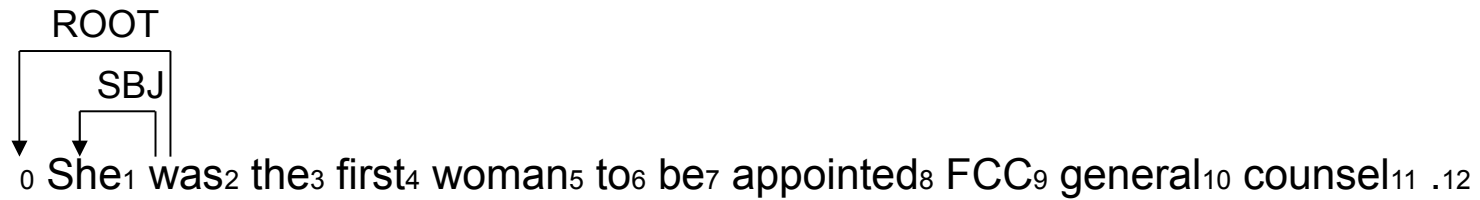
Example

[(1,2), (2,0)]

j: 4

i: 1

{l1=0.0, **shift**=0.9998}



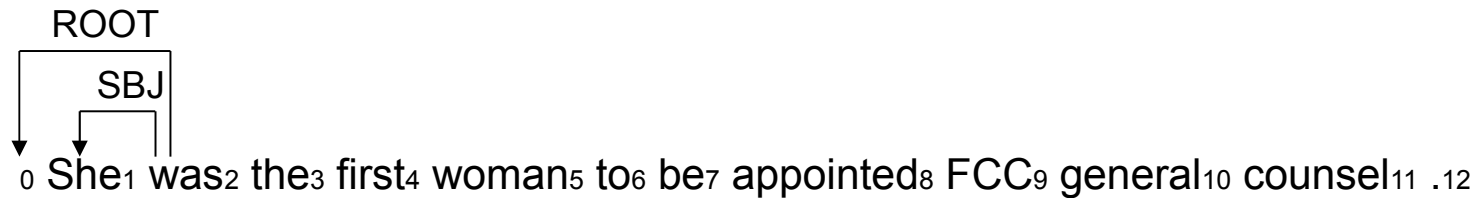
Fundamental Parsing Algorithm Example

[(1,2), (2,0)]

j: 4

i: 0

{|1=5.0E-4, **shift**=0.9995}



Fundamental Parsing Algorithm

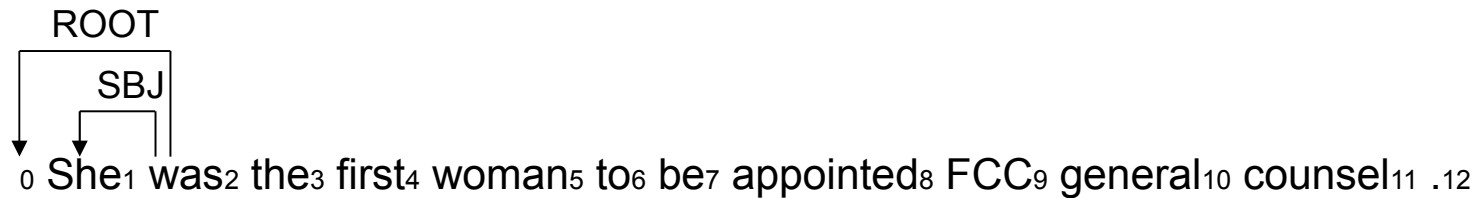
Example

[(1,2), (2,0)]

j: 5

i: 4

{I2=0.8812, I1=0.013, shift=0.1058}



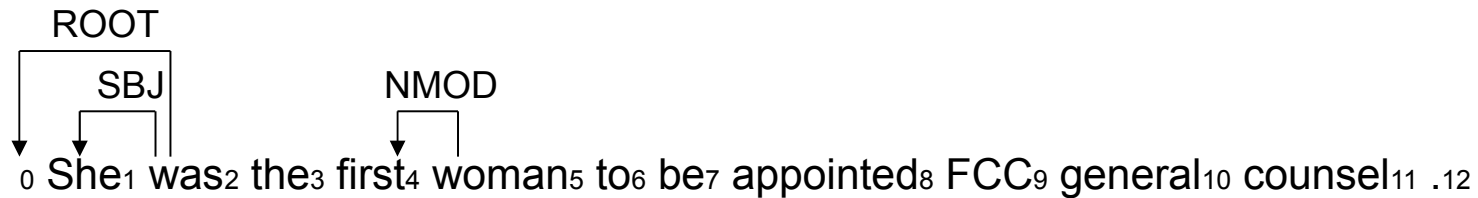
Fundamental Parsing Algorithm Example

[(1,2), (2,0), (5,4)]

j: 5

i: 3

{I2=0.8812, I1=0.013, shift=0.1058}



Fundamental Parsing Algorithm

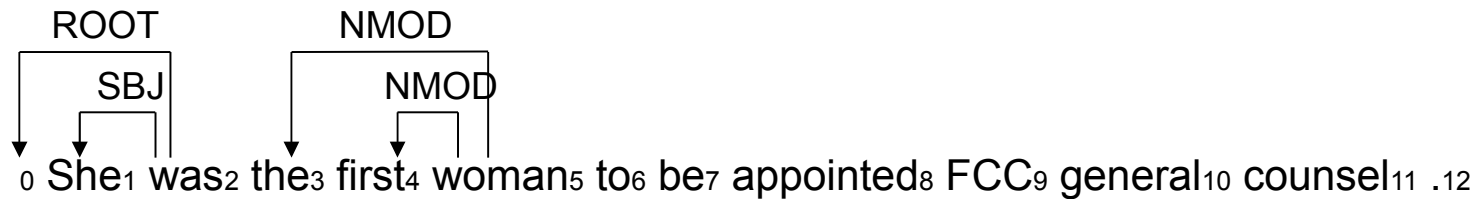
Example

[(1,2), (2,0), (5,4), (5,3)]

j: 5

i: 2

{I1=0.8364, shift=0.1415}



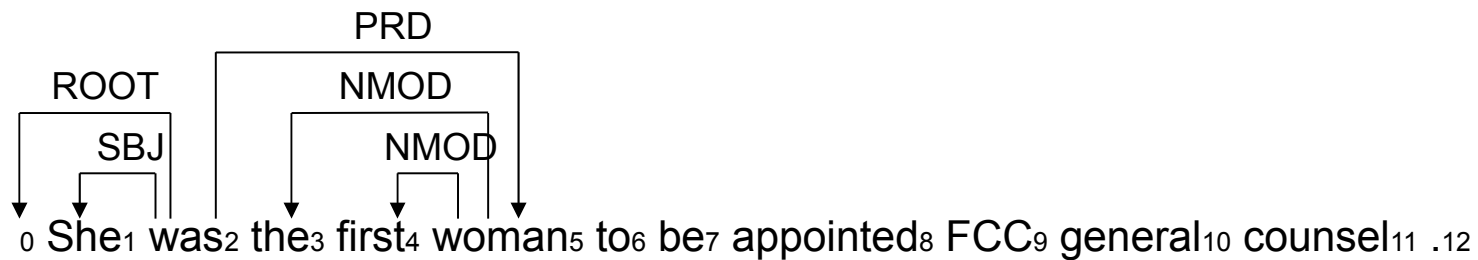
Fundamental Parsing Algorithm Example

[(1,2), (2,0), (5,4), (5,3), (2,5)]

j: 6

i: 5

{I1=0.6597, shift=0.2593}



Fundamental Parsing Algorithm

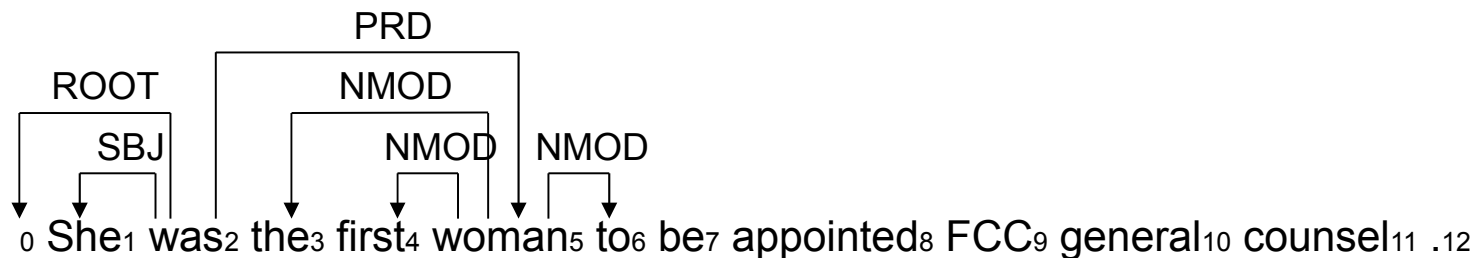
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5)]

j: 7

i: 6

{I1=0.9699, shift=0.0286}



Fundamental Parsing Algorithm

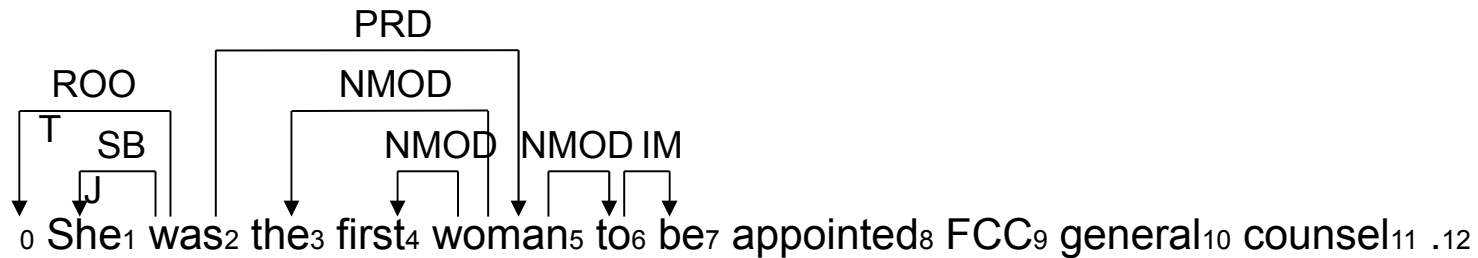
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6)]

j: 8

i: 7

{I1=0.4813, shift=0.5179}



Fundamental Parsing Algorithm

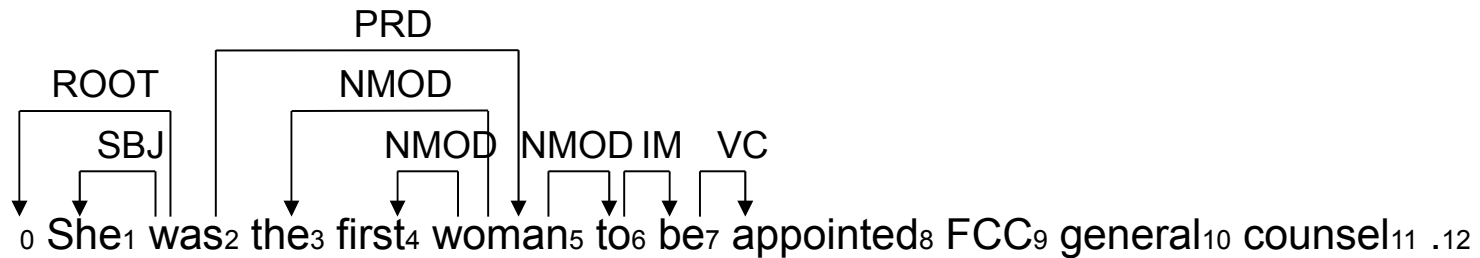
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 9

i: 8

{l1=0.0987, **shift**=0.8476}



Fundamental Parsing Algorithm

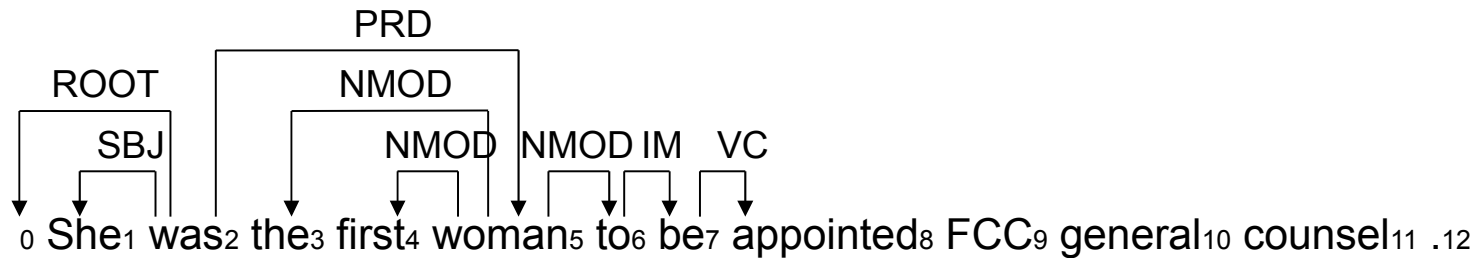
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 9

i: 7

{l1=0.0404, **shift**=0.9589}



Fundamental Parsing Algorithm

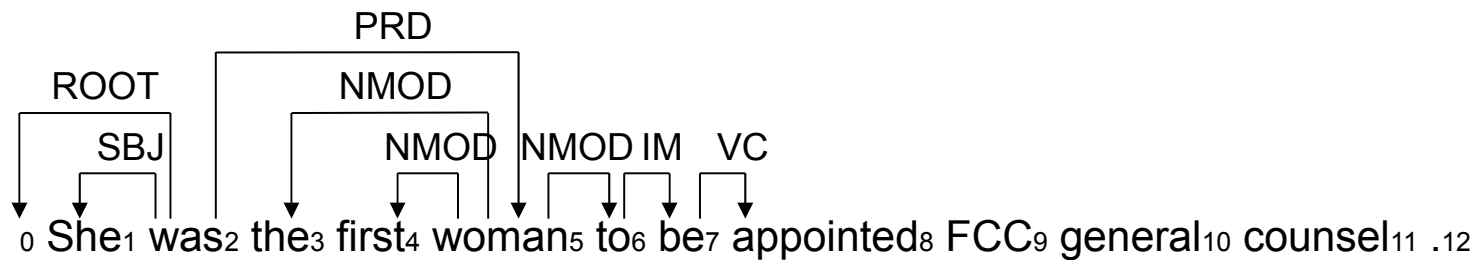
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 9

i: 6

{l1=0.0099, **shift**=0.9894}



Fundamental Parsing Algorithm

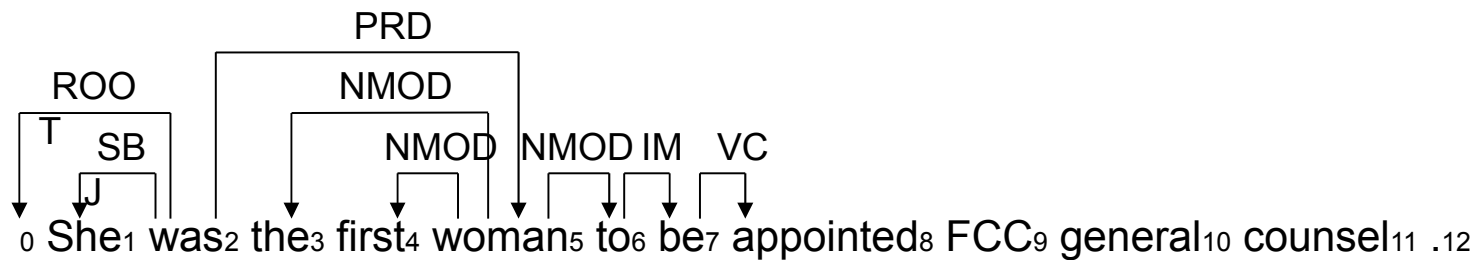
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 9

i: 4

{l1=0.0010, **shift**=0.9787}



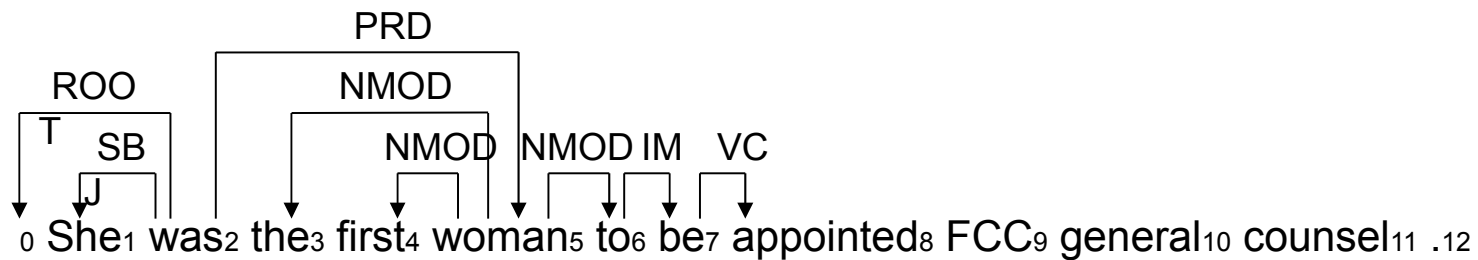
Fundamental Parsing Algorithm Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 9

i: 3

{l1=0.0, **shift**=0.9917}



Fundamental Parsing Algorithm

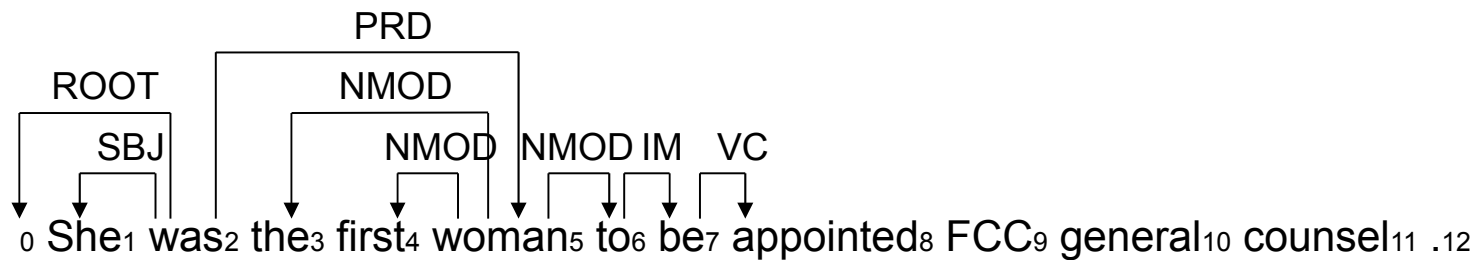
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 9

i: 2

{l1=0.0371, **shift**=0.9576}



Fundamental Parsing Algorithm

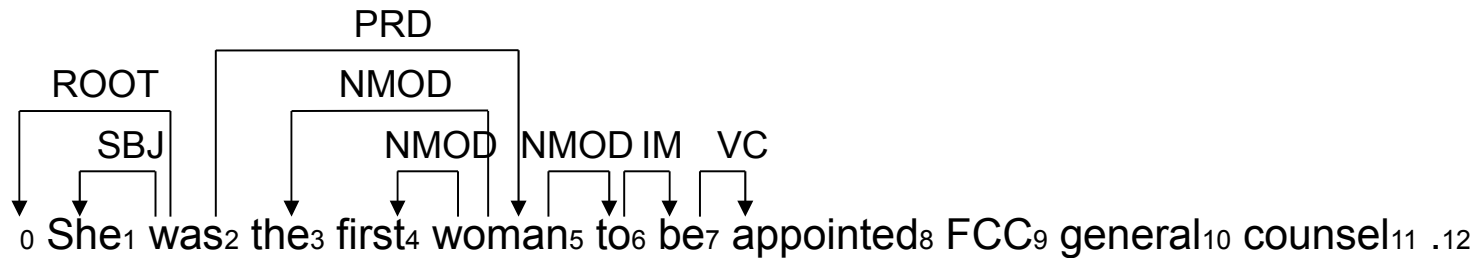
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 9

i: 1

{l1=2.0E-4, **shift**=0.9901}



Fundamental Parsing Algorithm

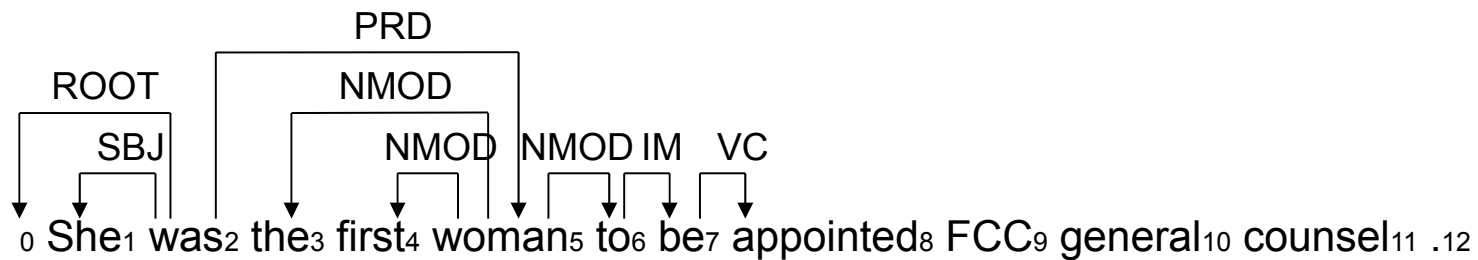
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 9

i: 0

{l1=0.0055, **shift**=0.9878}



Fundamental Parsing Algorithm

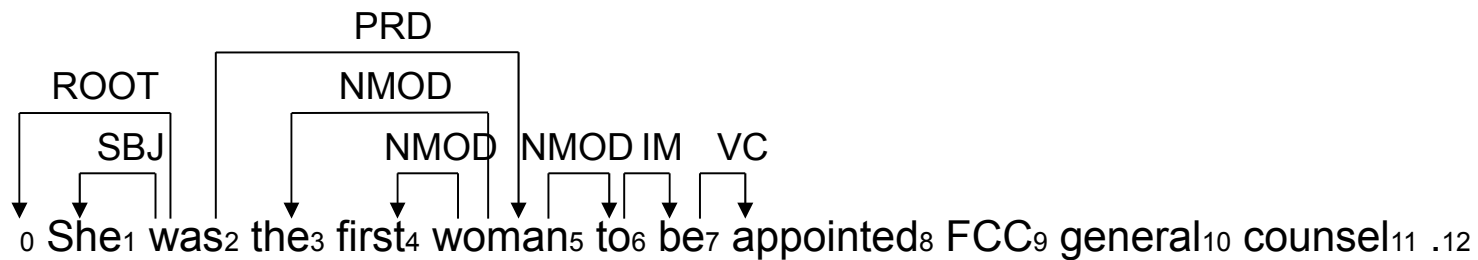
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 10

i: 9

{I2=0.1853, I1=0.0035, **shift**=0.8112}



Fundamental Parsing Algorithm

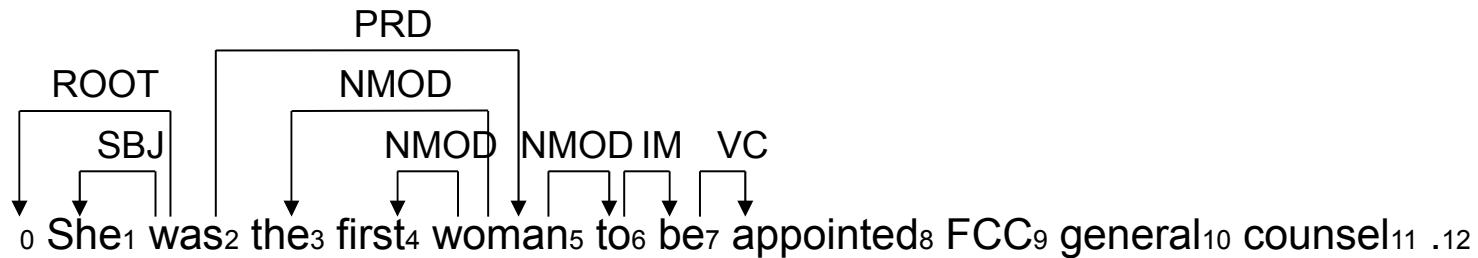
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 10

i: 8

{l1=0.0344, **shift**=0.9581}



Fundamental Parsing Algorithm

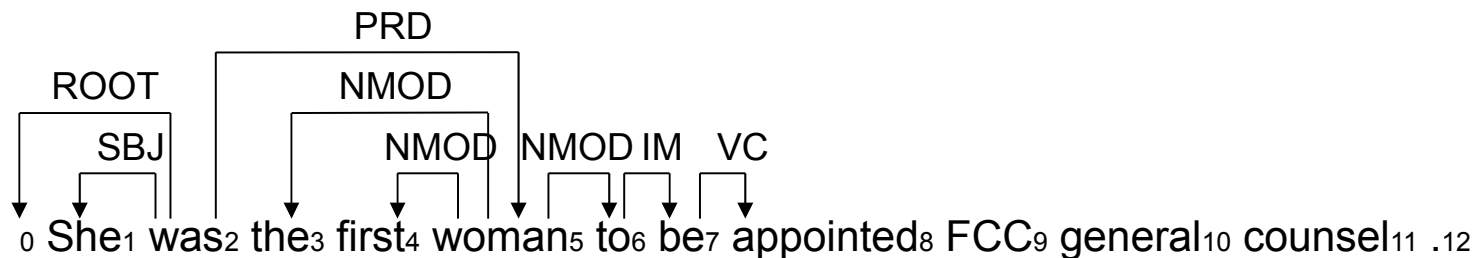
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 10

i: 7

{l1=0.0115, **shift**=0.9884}



Fundamental Parsing Algorithm

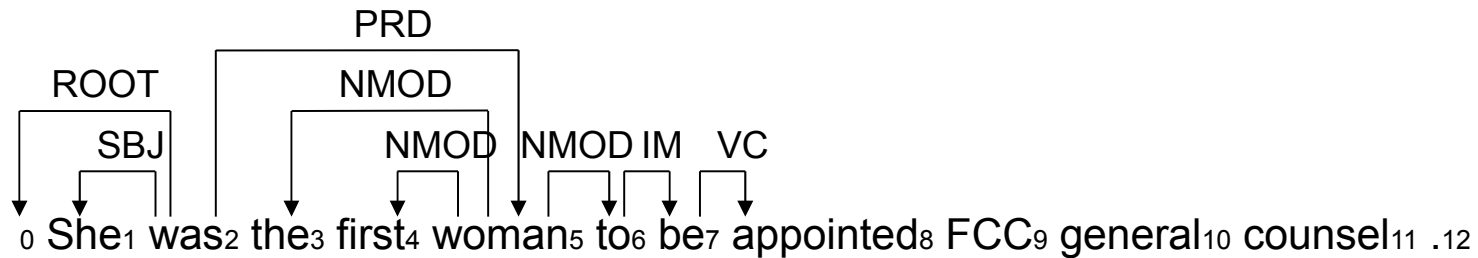
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 10

i: 6

{l1=0.0072, **shift**=0.9924}



Fundamental Parsing Algorithm

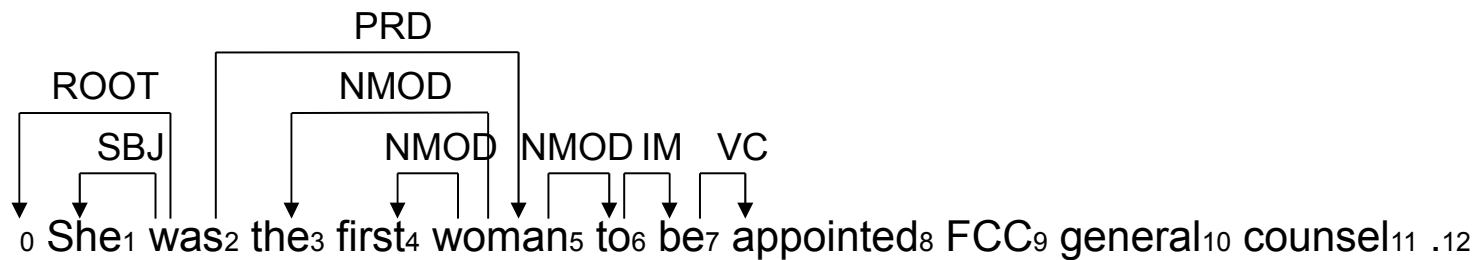
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 10

i: 5

{l1=0.0039, **shift**=0.9858}



Fundamental Parsing Algorithm

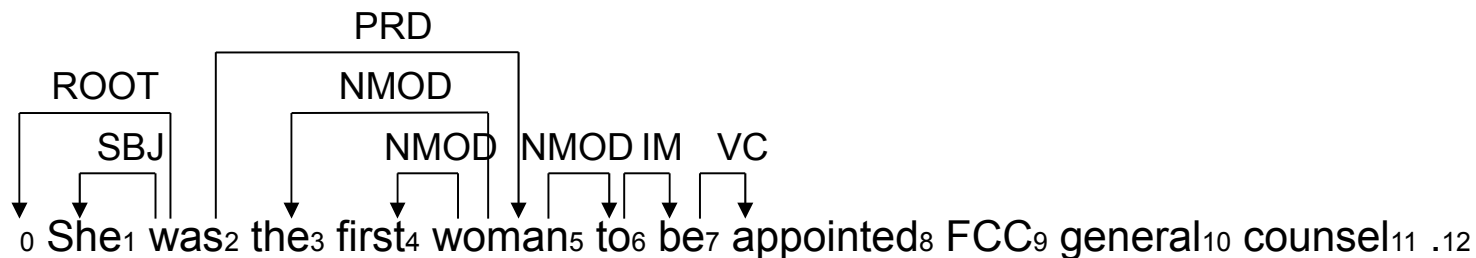
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 10

i: 4

{|1=4.0E-4, **shift**=0.9948}



Fundamental Parsing Algorithm

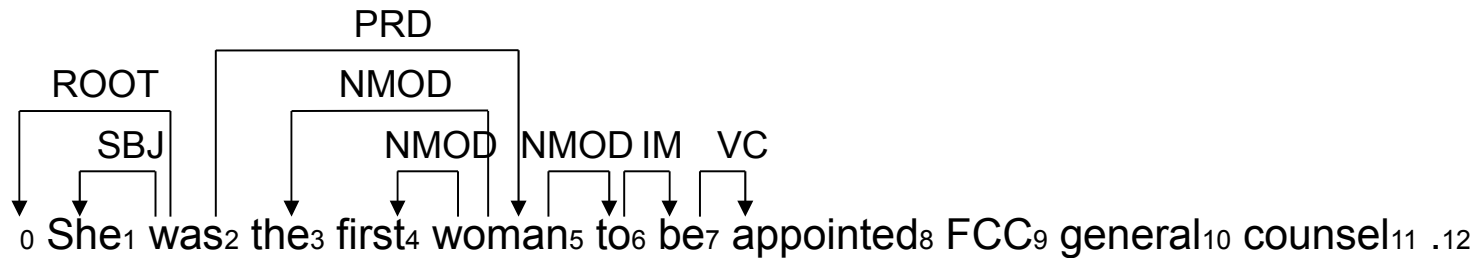
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 10

i: 2

{l1=0.0272, **shift**=0.9697}



Fundamental Parsing Algorithm

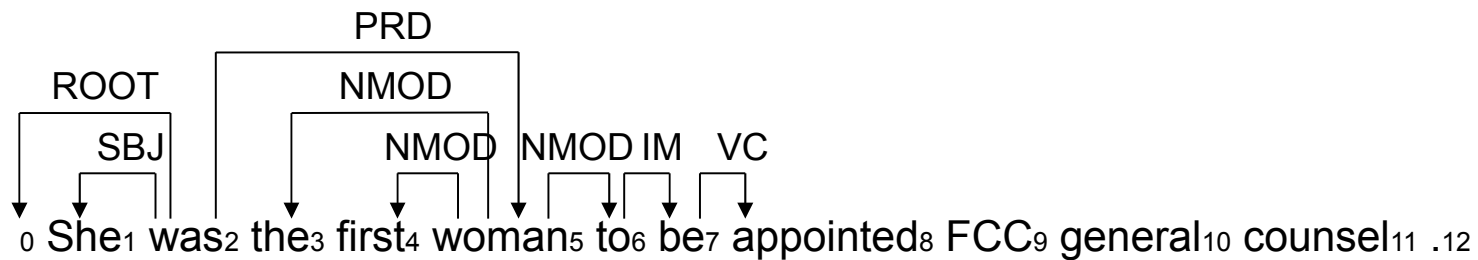
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 10

i: 1

{l1=2.0E-4, **shift**=0.9942}



Fundamental Parsing Algorithm

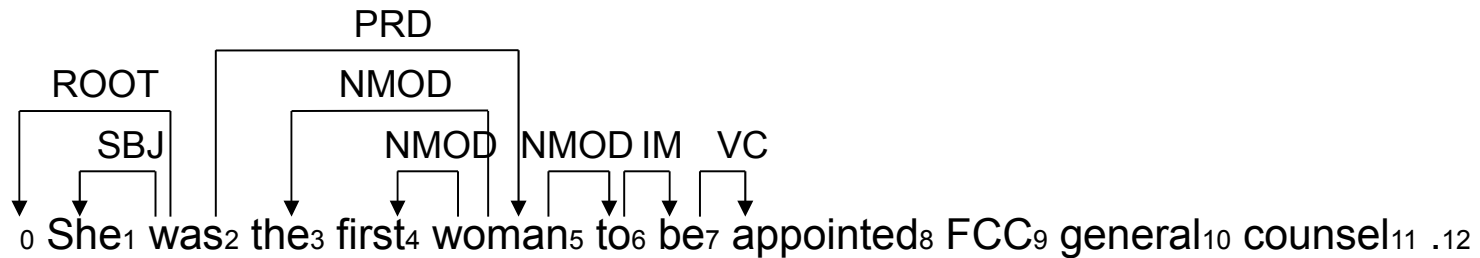
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 10

i: 0

{l1=0.0040, **shift**=0.9921}



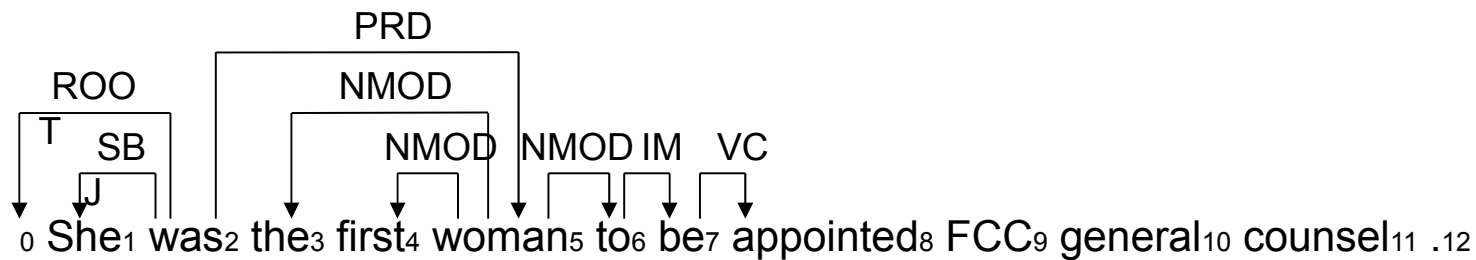
Fundamental Parsing Algorithm Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7)]

j: 11

i: 10

{I2=0.971, I1=0.0143, shift=0.0147}



Fundamental Parsing Algorithm

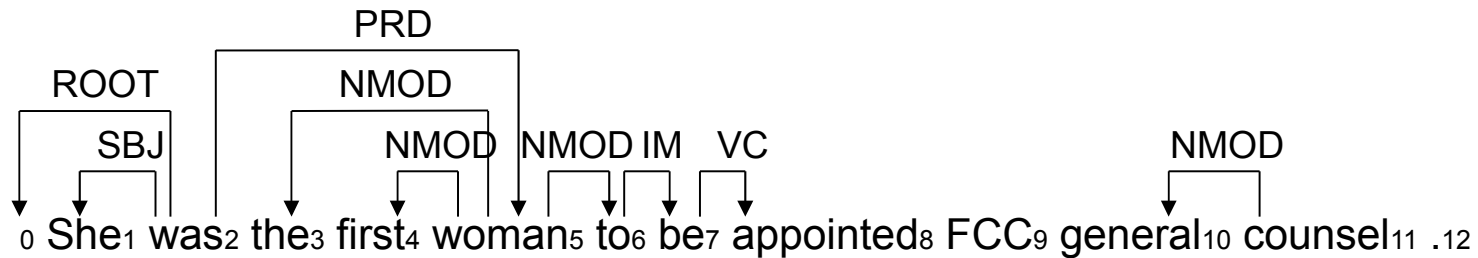
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7), (10,11)]

j: 11

i: 9

{I2=0.9516, I1=0.023, shift=0.0255}



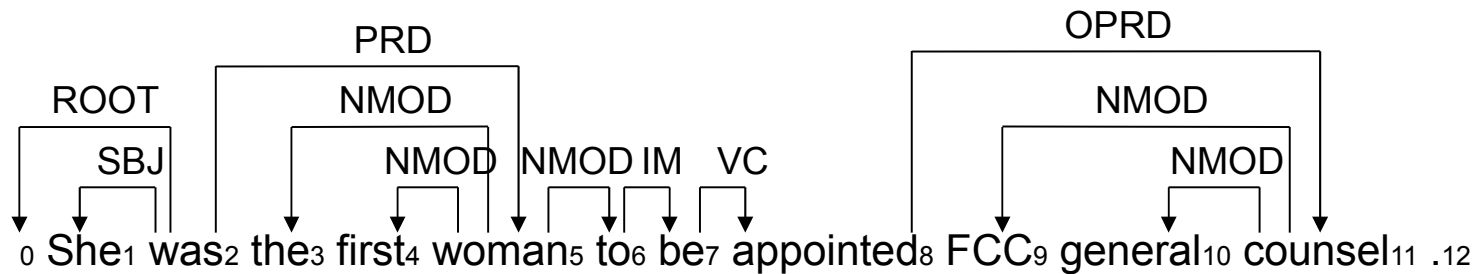
Fundamental Parsing Algorithm Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7), (10,11), (9,11), (11,8)]

j: 12

i: 11

{|1=0.1134, **shift**=0.8858}



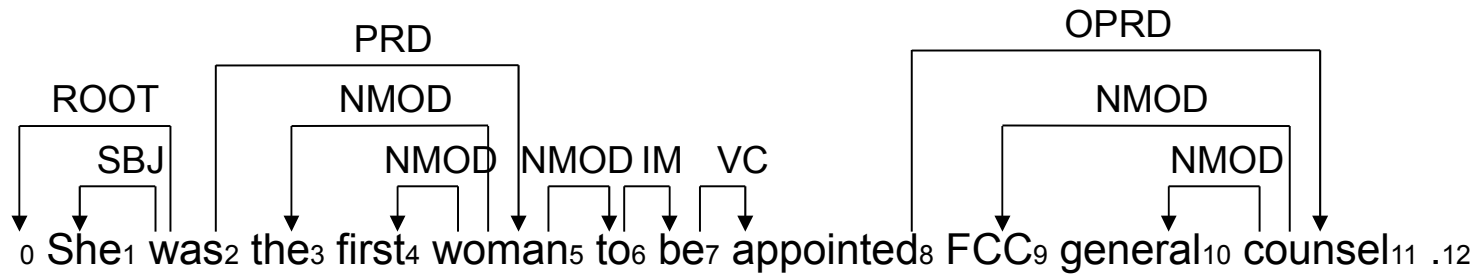
Fundamental Parsing Algorithm Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7), (10,11), (9,11), (11,8)]

j: 12

i: 10

{l1=0.0125, **shift**=0.9865}



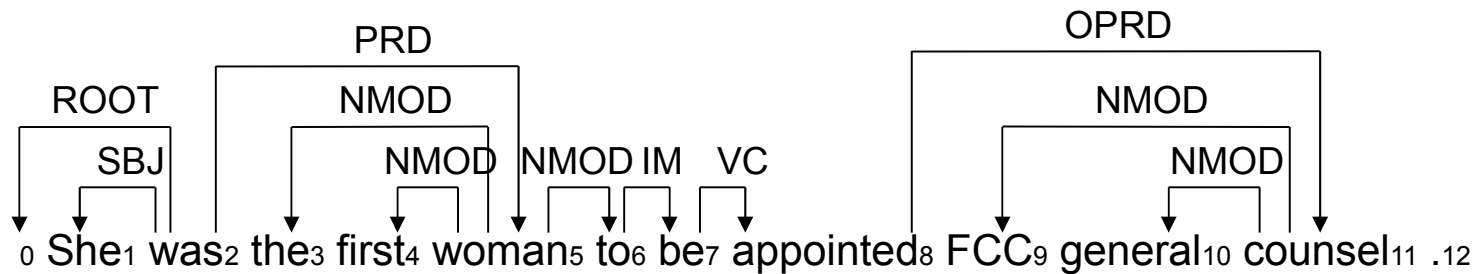
Fundamental Parsing Algorithm Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7), (10,11), (9,11), (11,8)]

j: 12

i: 9

{l1=0.0134, **shift**=0.9856}



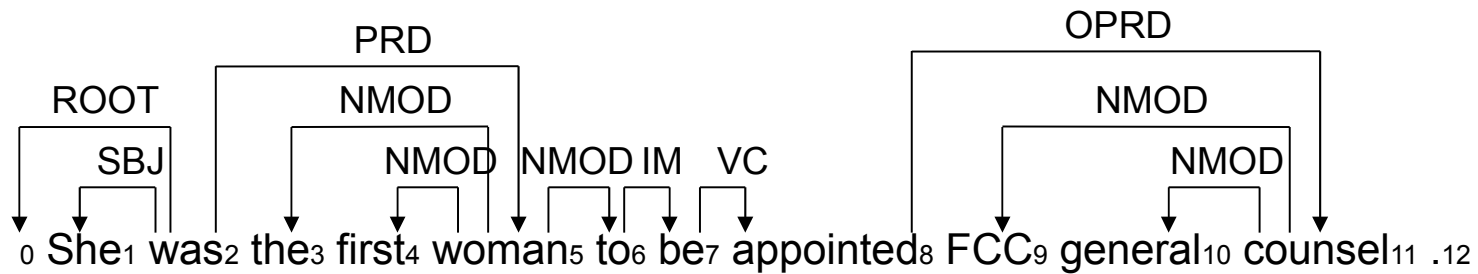
Fundamental Parsing Algorithm Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7), (10,11), (9,11), (11,8)]

j: 12

i: 8

{|1=0.2228, **shift**=0.7771}



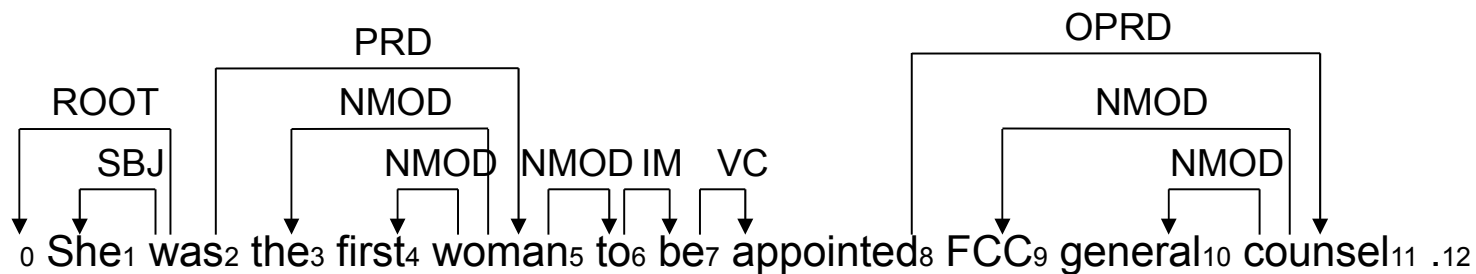
Fundamental Parsing Algorithm Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7), (10,11), (9,11), (11,8)]

j: 12

i: 7

{l1=0.2711, **shift**=0.7289}



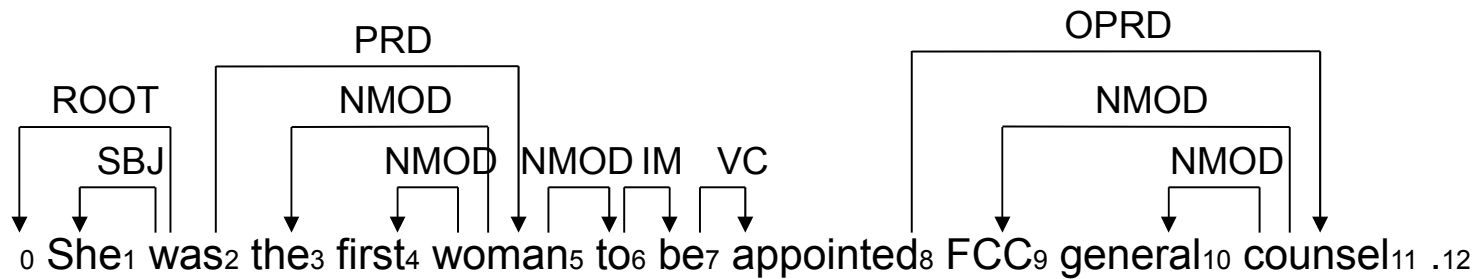
Fundamental Parsing Algorithm

Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7), (10,11), (9,11), (11,8)]

j: 12
i: 6

{l1=0.1105, **shift**=0.8895}



Fundamental Parsing Algorithm

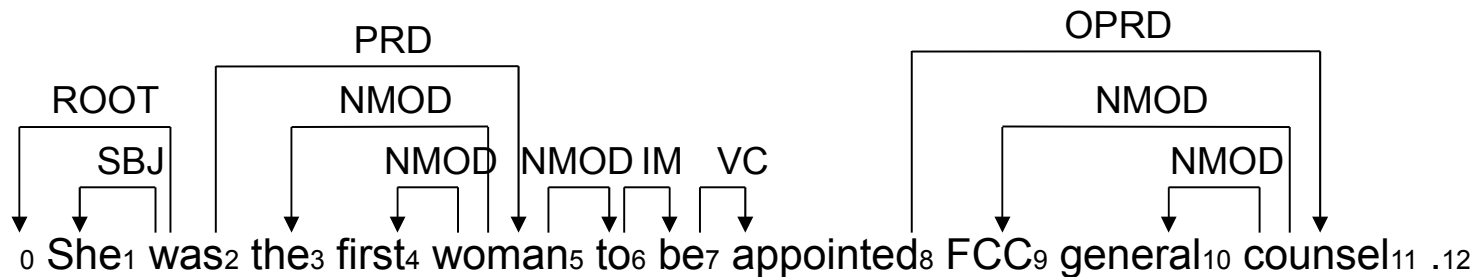
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7), (10,11), (9,11), (11,8)]

j: 12

i: 5

{l1=0.0583, **shift**=0.9415}



Fundamental Parsing Algorithm

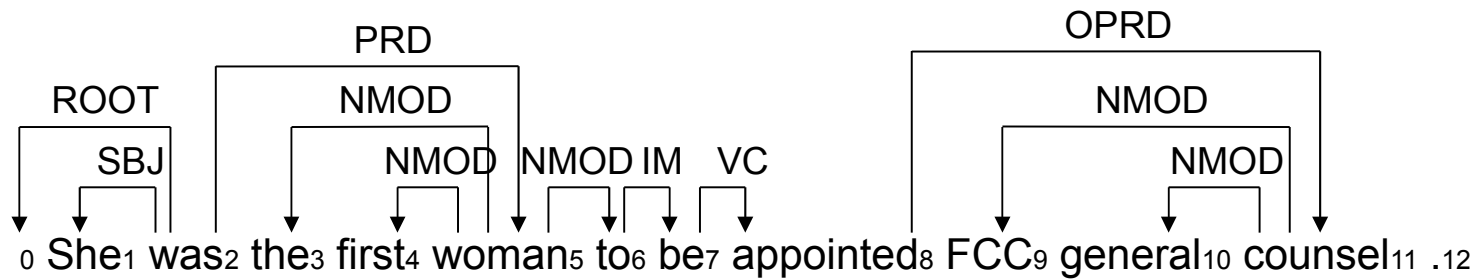
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7), (10,11), (9,11), (11,8)]

j: 12

i: 4

{l1=0.0154, **shift**=0.9843}



Fundamental Parsing Algorithm

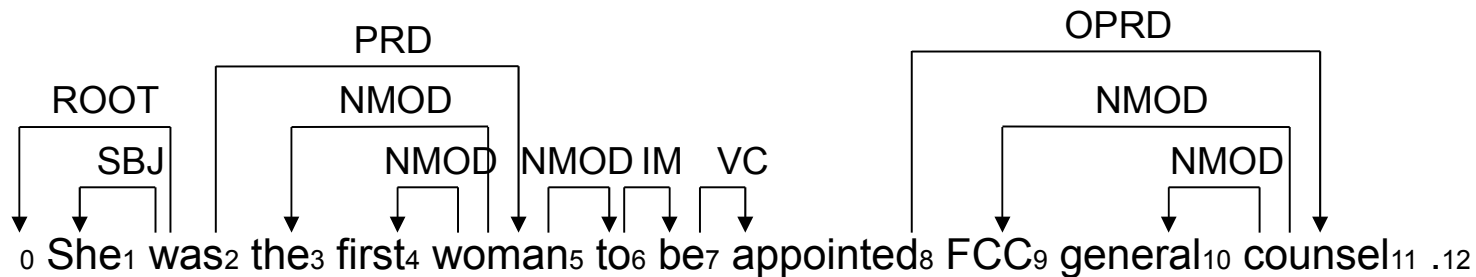
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7), (10,11), (9,11), (11,8)]

j: 12

i: 3

{l1=0.0, **shift**=0.9998}



Fundamental Parsing Algorithm

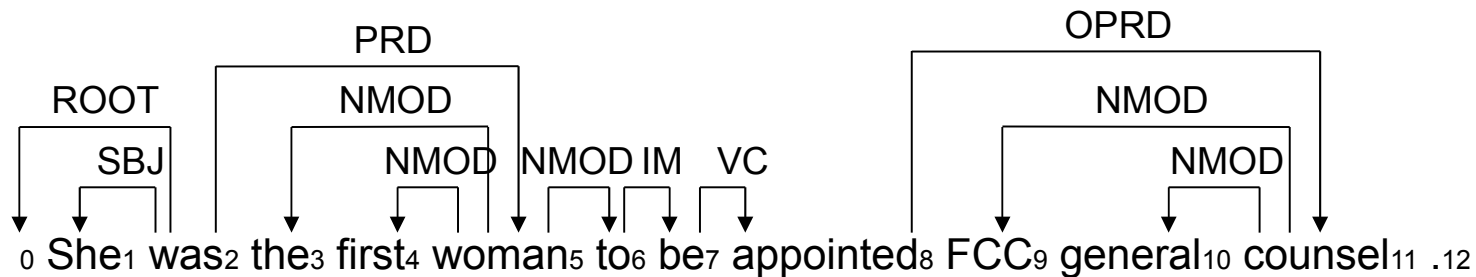
Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7), (10,11), (9,11), (11,8)]

j: 12

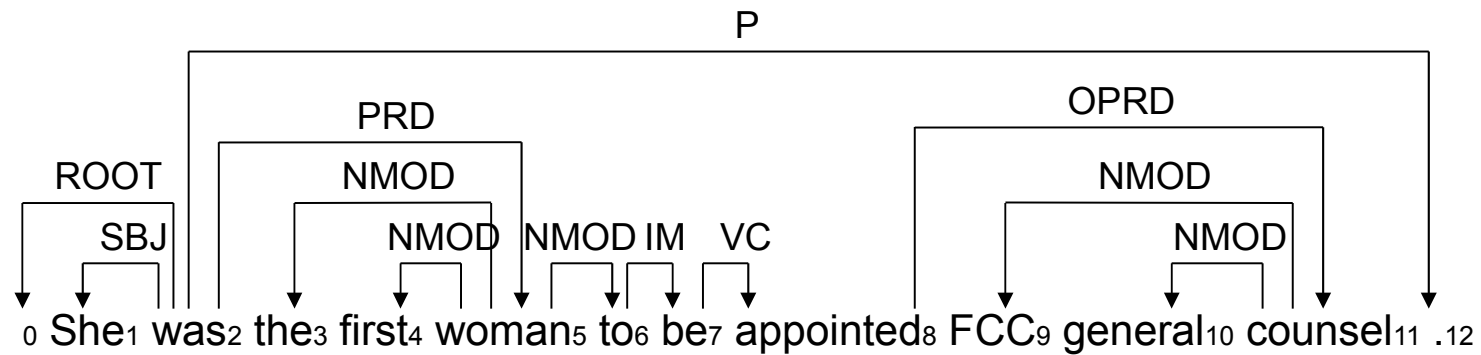
i: 2

{I1=0.4616, shift=0.5383}



Fundamental Parsing Algorithm Example

[(1,2), (2,0), (5,4), (5,3), (2,5), (6,5), (7,6), (8,7), (10,11), (9,11), (11,8), (12, 2)]



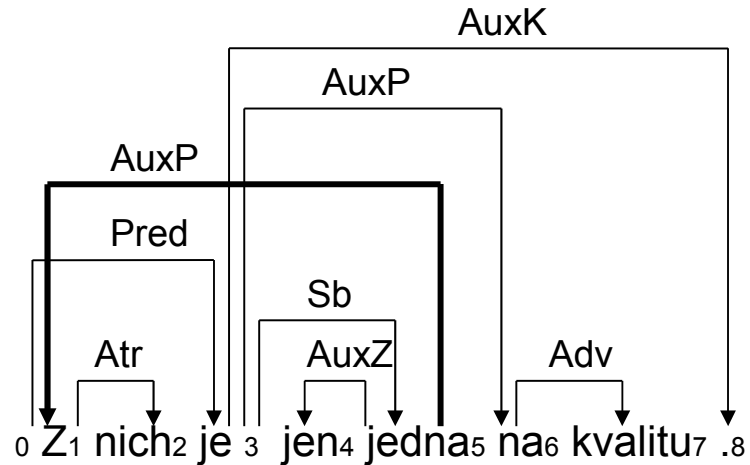
Output

1	She	PRP	2	SBJ
2	was	VBD	0	ROOT
3	the	DT	5	NMOD
4	first	JJ	5	NMOD
5	woman	NN	2	PRD
6	to	TO	5	NMOD
7	be	VB	6	IM
8	appointed	VBN	7	VC
9	FCC	NNP	11	NMOD
10	general	JJ	11	NMOD
11	counsel	NN	8	OPRD
12	.	.	2	P

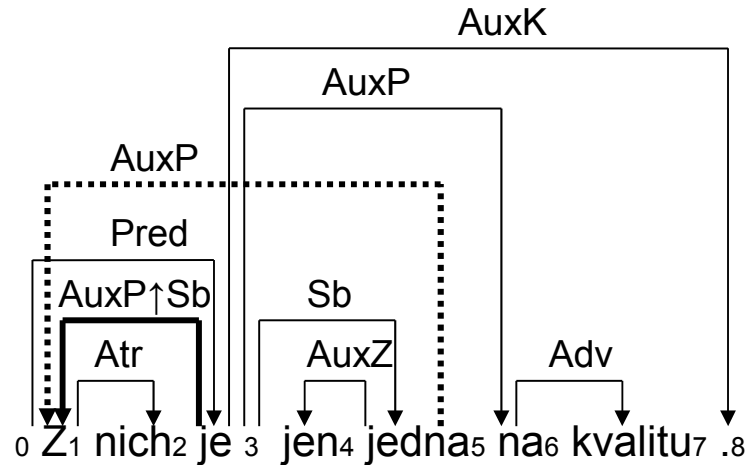
Pseudo-Projective Parsing

- Technique for projective parsers to create structures that can be transformed into non-projective structures
- Information about projectivity is encoded in arc labels

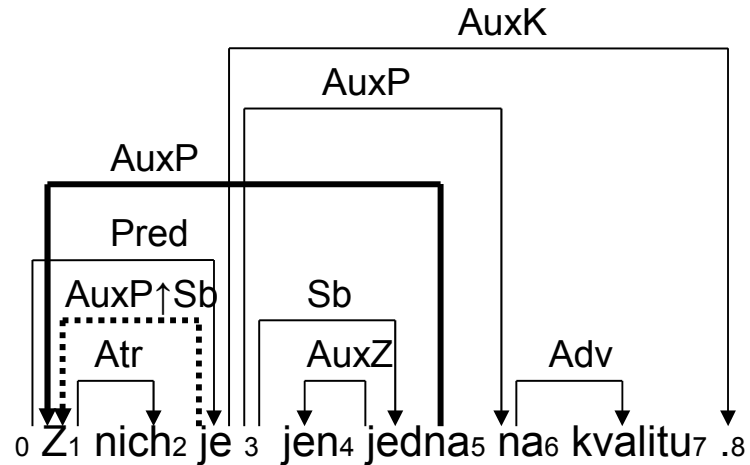
Pseudo-Projective Parsing Example



Pseudo-Projective Parsing Example



Pseudo-Projective Parsing Example



Machine Learning Libraries

- OpenNLP MaxEnt -
<http://maxent.sourceforge.net/howto.html>
- Mallet Toolkit - <http://mallet.cs.umass.edu/>
- Liblinear -
<http://www.csie.ntu.edu.tw/~cjlin/liblinear/>

OpenNLP

- OpenNLP – easiest tool, worst performance (accuracy and runtime)
- Two main classes: CreateModel and Predict
- Format:

- training instance:

Brendan N capitalized=true B-PER

- classification instance:

Richard N capitalized=true ?

- classification result:

B-PER[0.6441] I-PER[0.2060] 0[0.1499]

Example

Amerikaanse Adj B-MISC

tragikomische Adj O

film N O

van Prep O

Richard N B-PER

Benjamin N I-PER

uit Prep O

1996 Num O

met Prep O

Shirley N B-PER

MacLaine N I-PER

, Punc O

Ricki N B-PER

Lake N I-PER

, Punc O

Brendan N B-PER

Fraser N I-PER

, Punc O

Miguel N B-PER

Sandoval N I-PER

, Punc O

Loren N B-PER

Dean N I-PER

en Conj O

Peter N B-PER

Gerety N I-PER

. Punc O

Vorige Adj O

week N O

kondigde V O

minister N O

Aelvoet N B-PER

al Adv O

de Art O

invoering N O

van Prep O

Beltrace N B-MISC

(Punc O

Belgian N B-MISC

Tracability N I-MISC

) Punc O

aan Prep O

. Punc O

Vorige Adj ?

week N ?

kondigde V ?

minister N ?

Aelvoet N ?

al Adv ?

de Art ?

invoering N ?

van Prep ?

Beltrace N ?

(Punc ?

Belgian N ?

Tracability N ?

) Punc ?

aan Prep ?

. Punc ?

Mallet

- <http://mallet.cs.umass.edu/mallet-tutorial.pdf>
- Quite powerful, a little bit harder to use
- Important Classes:
 - Instance, InstanceList, Pipe
 - ClassifierTrainer, Classifier
 - Alphabet
 - Labeling
- Format: can be the same as with OpenNLP

Instances/Pipes/Alphabet

```
ArrayList<Pipe> pipeList = new ArrayList<Pipe>();

pipeList.add(new Target2Label());

pipeList.add(new CharSequence2TokenSequence());

pipeList.add(new TokenSequence2FeatureSequence());

pipeList.add(new FeatureSequence2FeatureVector());

Pipe pipe = new SerialPipes(pipeList);

InstanceList instances = new InstanceList(pipe);

FileInputStream in = new FileInputStream("input/ned.train");

InputStreamReader ir = new InputStreamReader(in, "UTF8");

BufferedReader fr = new BufferedReader(ir);

String line;

while ((line = fr.readLine()) != null) {

    if (line.length() > 0) {

        String features = line.substring(0, line.lastIndexOf(" "));

        String cl = line.substring(line.lastIndexOf(" ")+1);

        Instance i = new Instance(features, cl, null, null);

        instances.addThruPipe(i);

    }

}
```

Instance:

- Name - What is it called?
- **Data** - What is the input?
- **Target/Label** - What is the output?
- Source - What did it originally look like?

Pipes:

- Transform your input to internal representation
- Create alphabets

Alphabets:

- Allow efficient mappings
- Store features used during training
- Store possible class labels

Trainer/Classifier

```
MaxEntTrainer trainer = new MaxEntTrainer();
Classifier classifier = trainer.train(instances);
ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(classifierFile));
oos.writeObject(classifier);
oos.close();
```

```
ObjectInputStream serializedClassifier = new ObjectInputStream(new
FileInputStream(classifierFile));
Classifier classifier = (Classifier)serializedClassifier.readObject();
FileInputStream in = new FileInputStream("input/ned.test");
InputStreamReader ir = new InputStreamReader(in, "UTF8");
BufferedReader fr = new BufferedReader(ir);
Pipe classifierPipe = classifier.getInstancePipe();
instances = new InstanceList(classifierPipe);
while ((line = fr.readLine()) != null) {
    if (line.length() > 0) {
        String features = line.substring(0, line.lastIndexOf(" "));
        String cl = line.substring(line.lastIndexOf(" ") + 1);
        Instance i = new Instance(features, cl, null, null);
        instances.addThruPipe(i);
        System.out.println(classifier.classify(i).getLabeling().getBestLabel() + " " + cl);
    }
}
```

Liblinear

- Most powerful library, a little bit more effort to use
- You have to create and work with Alphabets yourself
- Format:

Amerikaanse Adj B-MISC	Amerikaanse=1 Adj=2 B-MISC=0	1:1 2:1 0
tragikomische Adj O	tragikomische=3 Adj=2 O=1	2:1 3:1 1
film N O	film=4 N=5 O=1	4:1 5:1 1
van Prep O	van=6 Prep=7 O=1	6:1 7:1 1

Models

```
private double bias = -1;

private Problem prob = null;

private Parameter param = new Parameter(SolverType.L1R_LR, 1, 0.01);

readProblem(trainingDataFile);

int l = this.prob.l;

int[] perm = new int[l];

GroupClassesReturn rv = groupClasses(this.prob, perm);

Model model = Linear.train(this.prob, this.param);

model.save(new File(modelFile));

Model.load(new File(modelFile));
```

groupClasses()

```
private static GroupClassesReturn groupClasses(Problem prob, int[] perm) {
    int l = prob.l;
    int max_nr_class = 16;
    int nr_class = 0;

    int[] label = new int[max_nr_class];
    int[] count = new int[max_nr_class];
    int[] data_label = new int[l];
    int i;

    for (i = 0; i < l; i++) {
        int this_label = prob.y[i];
        int j;
        for (j = 0; j < nr_class; j++) {
            if (this_label == label[j]) {
                ++count[j];
                break;
            }
        }
        data_label[i] = j;
        if (j == nr_class) {
            if (nr_class == max_nr_class) {
                max_nr_class *= 2;
                label = copyOf(label, max_nr_class);
                count = copyOf(count, max_nr_class);
            }
            label[nr_class] = this_label;
            count[nr_class] = 1;
            ++nr_class;
        }
    }

    int[] start = new int[nr_class];
    start[0] = 0;
    for (i = 1; i < nr_class; i++)
        start[i] = start[i - 1] + count[i - 1];
    for (i = 0; i < l; i++) {
        perm[start[data_label[i]]] = i;
        ++start[data_label[i]];
    }
    start[0] = 0;
    for (i = 1; i < nr_class; i++)
        start[i] = start[i - 1] + count[i - 1];

    return new GroupClassesReturn(nr_class, label, start, count);
}
```

readProblem()

```
public void readProblem(String filename) throws IOException, InvalidInputDataException {
    BufferedReader fp = new BufferedReader(new FileReader(filename));
    List<Integer> vy = new ArrayList<Integer>();
    List<FeatureNode[]> vx = new ArrayList<FeatureNode[]>();
    int max_index = 0;
    int lineNr = 0;

    try {
        while (true) {
            String line = fp.readLine();
            if (line == null) break;
            lineNr++;

            StringTokenizer st = new StringTokenizer(line, "\t\n\r\f:");
            String token = st.nextToken();

            try {
                vy.add(liblinear.Linear.atoi(token));
            } catch (NumberFormatException e) {
                throw new InvalidInputDataException("invalid label: " + token, filename, lineNr, e);
            }

            int m = st.countTokens() / 2;
            FeatureNode[] x;
            if (bias >= 0) {
                x = new FeatureNode[m + 1];
            } else {
                x = new FeatureNode[m];
            }
            int indexBefore = 0;
            for (int j = 0; j < m; j++) {
                token = st.nextToken();
                int index;
                try {
                    index = liblinear.Linear.atoi(token);
                } catch (NumberFormatException e) {
                    throw new InvalidInputDataException("invalid index: " + token, filename, lineNr, e);
                }

                // assert that indices are valid and sorted
                if (index < 0) throw new InvalidInputDataException("invalid index: " + index, filename, lineNr);
                if (index <= indexBefore) throw new InvalidInputDataException("indices must be sorted in ascending order", filename, lineNr);
                indexBefore = index;
            }
            token = st.nextToken();
            try {
                double value = liblinear.Linear.atof(token);
                x[j] = new FeatureNode(index, value);
            } catch (NumberFormatException e) {
                throw new InvalidInputDataException("invalid value: " + token, filename, lineNr);
            }
        }
        if (m > 0) {
            max_index = Math.max(max_index, x[m - 1].index);
        }

        vx.add(x);
    }
    prob = constructProblem(vy, vx, max_index);
}
finally {
    fp.close();
}
```

Classification

```
String[] instance = integerRepresentation.split(" ");
FeatureNode[] predArray = new FeatureNode[instance.length-1];
double[] scoreArray = new double[alphabet.getNumberOfLabels()];
// calculating the score for each label
for (int i=1; i < instance.length; i++) {
    String[] fArray = instance[i].split(":");
    int fIndex = Integer.valueOf(fArray[0]);
    if (scoreArray.length != 2) {
        for (int k=0; k < scoreArray.length; k++) {
            double w = model.w[(fIndex-1)*scoreArray.length+k];
            scoreArray[k] += w;
        }
    }
    else {
        scoreArray[0] += model.w[fIndex-1];
        scoreArray[1] += (-1)*model.w[fIndex-1];
    }
    FeatureNode fn = new FeatureNode(fIndex,1);
    predArray[i-1] = fn;
}
predictedLabelIndex = -1;
double max = Double.NEGATIVE_INFINITY;
double[] probArray = new double[scoreArray.length];
// determining the index of the top scored label
for (int i=0; i < scoreArray.length; i++) {
    if (scoreArray[i] > max ) {
        max = scoreArray[i];
        predictedLabelIndex = i+1;
    }
}
// calculating probabilities out of the scores
double sum = 0.0;
for (int i=0; i < probArray.length; i++) {
    probArray[i] = Math.exp(scoreArray[i]-max);
    sum += probArray[i];
}
// normalization
for (int i=0; i < probArray.length; i++) {
    probArray[i] /= sum;
}
}
```

1:1 3:1 5:1 9:1

Alphabet(Labels) = {0,1,2}

0.5 1.5 -2.0

0.3 0.2 -0.5

0 1.0 -1.0

0.2 -0.1 -0.1

0.4

0.2

-0.3

-0.1

Corresponds to the
Classification slide from Lecture 1

CoNLL Format

1 – Index, 2 – Word Form, 3 – Lemma,
4 – POS(coarse), 5 – POS(fine), 6 – Morphology,
7 – Head, 8 – Dep type

```
1 Ms. Ms. NNP NNP _ 2 TITLE
2 Haag Haag NNP NNP _ 3 SBJ
3 plays plays VBZVBZ_ 0 ROOT
4 Elianti Elianti NNP NNP _ 3 OBJ
5 . . . . _ 3 P
```

Enjoy training your own dependency parser!