TEAM DJANGO [Project Report]

Ayan Biswas - 2019121009
Veeral Agarwal - 2019114009
Mentor - Saujas Vaduguru
Course instructor - prof. Manish Shrivastava

Dependency Parsing -[Hindi]

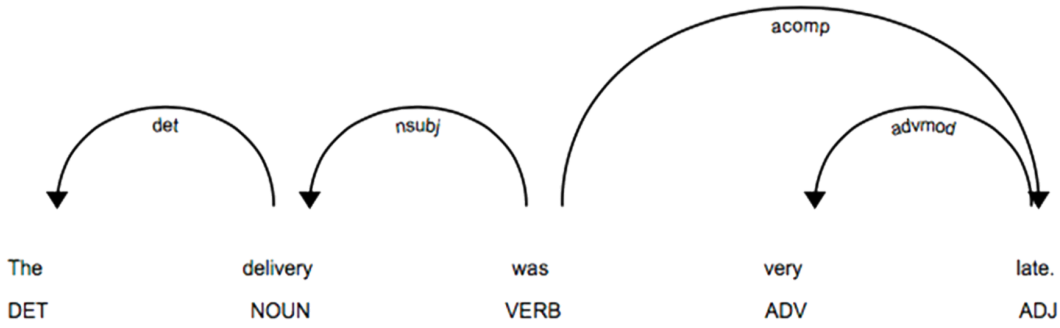
April 30, 2021

Problem Statement

In this project, we are creating a dependency parser (Hindi) for which the training dataset will be provided. A dependency parser identifies the syntactic dependency between words in a sentence. While a lot of literature exists in the dependency parser community in NLP, there are few well-developed tools for Indian languages.

Introduction

A dependency parser explores a sentence's grammatical form, defining associations between "head" words and modifier words. A dependency parse of a short sentence is seen in the diagram below. The arrow from moving to faster shows that faster modify moving, and the mark ad mod applied to the arrow defines the dependency's exact existence.



Transition based parsing:

This parser produces a parse by scanning the terms of a sentence in linear time. It holds a partial parse, a stack of words currently being processed, and a buffer of words yet to be processed at each stage. Once the buffer is complete, the parser proceeds to add transformations to its state.

The initial state is to have all of the words in order on the buffer, with a single dummy ROOT node on the stack. The following transitions can be applied:

LEFT-ARC marks the second item on the stack as a dependent of the first item and removes the second item from the stack (if the stack contains at least two items).

RIGHT-ARC marks the first item on the stack as a dependent of the second item and removes the first item from the stack (if the stack contains at least two items).

SHIFT: removes a word from the buffer and pushes it onto the stack (if the buffer is not empty).

A parser can produce any projective dependency parse using only these three forms of transformations. It's worth mentioning that in a typed dependence parser, we must also define the form of the relationship between the head and dependent being represented for each change.

A neural network classifier is used by the parser to select between transformations at each state. This classifier receives distributed representations (dense, continuous vector representations) of the parser's current state as inputs, and then selects between the potential transformations to render next. In the parser state, these representations identify different facets of the current stack and buffer contents.

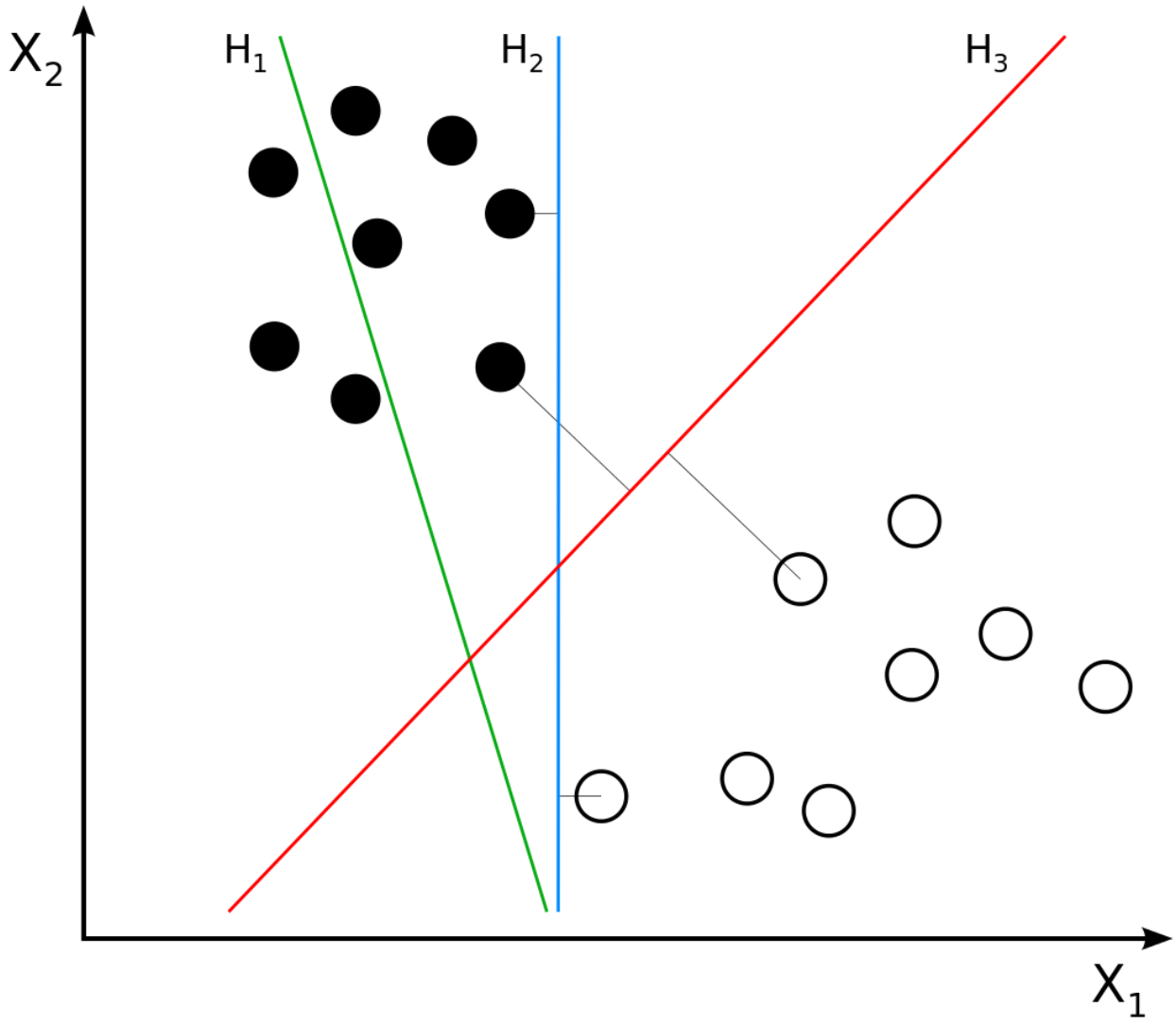
Methods

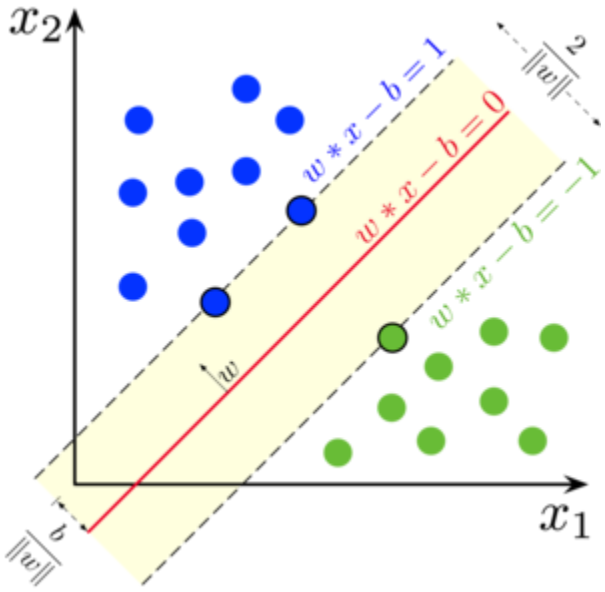
Greedy Transition/Shift Parsing - Greedy choice to be done using a Learning Algorithm

- SVMs / Neural networks
- SVMs are comparatively simple, we can take a feature vector and use kernels.
- Neural Nets would require us to take a look at the behavior of a few different architectures.

We are using SVMs for our project. A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems.

More formally, a support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.


X2
H1
H2
H3
X1



Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

Dataset

http://ltrc.iiit.ac.in/treebank_H2014/

We have used the data corpus taken from LTRC, IIIT Hyderabad. The data that was provided was already a parsed data by hindi shallow parser in Shakti Standard Form(SSF). This parsed data was used for training , testing and development.

Here is an illustration of how the parsed sentence of Hindi (in SSF structure) looks like and what all extraordinary syntactic and morphological data it gives.

<Sentence id='1'>

1 ((NP <fs name='NP' drel='nmod:NP2'>

1.1 यहाँ PRP <fs af='यहाँ,pn,,,,,o,,' name='यहाँ' posn='10'>

1.2 से PSP <fs af='से,psp,,,,,,' name='से' posn='20'>

))

2 ((NP <fs name='NP2' drel='jjmod:JJP'>

2.1 5 QC <fs af='5,num,any,any,,any,,' name='5' posn='30'>

2.2 किमी NN <fs af='किमी,n,m,sg,3,d,0,0' name='किमी' posn='40'>

2.3 दूरी NN <fs af='दूरी,n,f,sg,3,o,0,0' name='दूरी' posn='50'>

2.4 पर PSP <fs af='पर,psp,,,,,,' name='पर' posn='60'>

))

3 ((JJP <fs name='JJP' drel='nmod:NP3'>

3.1 स्थित JJ <fs af='स्थित,adj,any,any,,d,,' name='स्थित' posn='70'>

))

4 ((NP <fs name='NP3' drel='k1:VGF'>

4.1 वासुकि NNPC <fs af='वासुकि,n,m,sg,3,d,0,0' name='वासुकि' posn='80'>

4.2 ताल NNP <fs af='ताल,n,m,sg,3,d,0,0' name='ताल' posn='90'>

))

5 ((NP <fs name='NP4' drel='r6:NP5'>

5.1 अपने PRP <fs af='अपना,pn,m,any,any,o,0,0' name='अपने' posn='100'>

))

6 ((NP <fs name='NP5' drel='ccof:CCP'>

6.1 पारदर्शी JJ <fs af='पारदर्शी,adj,any,any,,o,,' name='पारदर्शी' posn='110'>

6.2 जल NN <fs af='जल,n,m,sg,3,o,0,0' name='जल' posn='120'>

))

7 ((CCP <fs name='CCP' drel='rt:VGF'>

7.1 और CC <fs af='और,avy,,,,,' name='और' posn='130'>

))

8 ((NP <fs name='NP6' drel='k7:VGNF'>

8.1 उसमें PRP <fs af='वह,pn,any,sg,3,o,में,meM' name='उसमें' posn='140'>

))

9 ((VGNF <fs name='VGNF' drel='nmod__k1inv:NP7'>

9.1 डूबते VMC <fs af='डूब,v,m,pl,any,,ता,wA' name='डूबते' posn='150'>

9.2 - SYM <fs af='-,punc,,,,,' name='-' posn='160'>

9.3 उतराते VM <fs af='उतरा,v,m,pl,any,,ता,wA' name='उतराते' posn='170'>

))

10 ((NP <fs name='NP7' drel='r6:NP8'>

10.1 हिमखंडों NN <fs af='हिमखंड,n,m,pl,3,o,0,0' name='हिमखंडों' posn='180'>

10.2 के PSP <fs af='का,psp,m,pl,,o,,' name='के' posn='190'>

))

11 ((NP <fs name='NP8' drel='ccof:CCP'>

11.1 अद्भुत JJ <fs af='अद्भुत,adj,any,any,,o,,' name='अद्भुत' posn='200'>

11.2 दृश्यों NN <fs af='दृश्य,n,m,pl,3,o,0,0' name='दृश्यों' posn='210'>

11.3 के PSP <fs af='के,psp,,,,,' name='के2' posn='220'>

11.4 लिए PSP <fs af='लिए,psp,,,,,' name='लिए' posn='230'>

))

12 ((JJP <fs name='JJP2' drel='k1s:VGF'>

12.1 विख्यात JJ <fs af='विख्यात,adj,any,any,,,,' name='विख्यात' posn='240'>

))

13 ((VGF <fs name='VGF' stype='declarative' voicetype='active'>

13.1 है VM <fs af='है,v,any,sg,3,,है,hE' name='है' posn='250'>

))

14 ((BLK <fs name='BLK' drel='rsym:VGF'>

14.1 । SYM <fs af='।,punc,,,,,' name='।' posn='260'>

))

</Sentence>

Dataset explanation

Beginning of each sentence is set apart with <Sentence id='x'> , where x is the id of the relating sentence which will be utilized as the sole file for finding a sentence. Moreover sentence finishing is set apart by </Sentence>. The sentence is partitioned into pieces/chunks (the above sentence has 14 chunks). Each chunk has a piece tag, chunk name. Each chunk has a drel segment which has two components isolated by a colon(:) . The subsequent component is the name of the chunk whose subordinate is the current chunk and the principal component is the reliance relationship between the two chunk. There is also the morphological information about each word in the context.

Our approach/procedure

Our approach to solve this problem includes nine steps.

1. Collecting the data with some manual cleaning.
2. Simplifying(data which includes only those informations which we need later) that data.
3. Head extraction.

4. Dependencies extraction.
5. Finding the indices which are not parsable.
6. Removing non parsable sentences.
7. Creating some unknown dependencies.
8. Creating models using different features.
9. Testing on models.

Data extraction and simplification

The interaction of Data Extraction just methods extricating just that information that was valuable for us in the further cycles and addressing them in a less difficult manner for simple availability. The arrangement of a sentence in the SSF format was changed to an easier structure with just that segment of the information that was helpful.

The simplified data is now as follows -

<Sentence id='1'>

H NP NP nmod NP2

T यहाँ PRP यहाँ

T से PSP से

H NP NP2 jjmod JJP

T 5 QC 5

T किमी NN किमी

T दूरी NN दूरी

T पर PSP पर

H JJP JJP nmod NP3

T स्थित JJ स्थित

H NP NP3 k1 VGF

T वासुकि NNPC वासुकि

T ताल NNP ताल

The lines starting with ‘H’ marks the chunk start and the lines starting with ‘T’ marks the words in the enclosing chunk.

Head extraction

We have nine different chunk tags - JJP, VGF, VGNN, VGNF, NEGP, RBP, BLK, CCP, NP

Noticing all various sorts of arrangements of words that establish a tag, and applying some etymology rules, we shaped principles for discovering the head word for each chunk tag.

Rules :

JJP: first find [‘JJ’] else ['QF','QC','QO']

VGNF: find [‘VM’]

VGF: find [‘VM’]

VGNN: find [‘VM’]

NEGP: find [‘NEG’]

RBP: find [‘RB’] else ['NN','WQ']

BLK: find ['SYM'] else ['UNK','RP','INJ']

CCP: find [‘CC’] else ['SYM']

NP: find ['NN','PRP','NNP'] else ['QC','QF','QO'] else ['NST'] else ['WQ']

For each chunk , emphasize in reverse in the rundown of words around there, and the main word that has its pos tag in the rundown of labels which had the principal inclination for that chunk tag is picked as the head. If head cannot be found then follow the same procedure for next

preferences. Further those sentences which were not following any stated rule were removed from the dataset.

Sentence format now -

<Sentence id='1'>

यहाँ से 5 किमी दूरी पर स्थित वासुकि ताल अपने पारदर्शी जल और उसमें डूबते - उतराते हिमखंडों के अद्भुत दृश्यों के लिए विख्यात है ।

यहाँ से 5 किमी दूरी पर स्थित वासुकि ताल अपना पारदर्शी जल और वह डूब - उतरा हिमखंड का अद्भुत दृश्य के लिए विख्यात है ।

PRP PSP QC NN NN PSP JJ NNPC NNP PRP JJ NN CC PRP VMC SYM VM NN PSP JJ NN PSP PSP JJ VM SYM

यहाँ दूरी स्थित ताल अपने जल और उसमें उतराते हिमखंडों दृश्यों विख्यात है ।

यहाँ दूरी स्थित ताल अपना जल और वह उतरा हिमखंड दृश्य विख्यात है ।

NP NP JJP NP NP NP CCP NP VGNF NP NP JJP VGF BLK

PRP NN JJ NNP PRP NN CC PRP VM NN NN JJ VM SYM

NP NP2 JJP NP3 NP4 NP5 CCP NP6 VGNF NP7 NP8 JJP2 VGF BLK

H यहाँ यहाँ NP PRP NP nmod NP2

H दूरी दूरी NP NN NP2 jjmod JJP

H स्थित स्थित JJP JJ JJP nmod NP3

H ताल ताल NP NNP NP3 k1 VGF

H अपने अपना NP PRP NP4 r6 NP5

H जल जल NP NN NP5 ccof CCP

The first word after ‘H’ is the word chosen as the head of the chunk, the next word is its lemmatized/root form of the word followed by the chunk tag, then the pos tag of the word chosen as the head.

Non parsable sentences [find and remove]

This assignment was completed to check the parsability of the sentences. As one doesn't need his model to prepare on some unacceptable dataset, we needed to eliminate each one of those sentences which were non parsable as in the sentences which were parsed wrong by the shallow hindi parser. Likewise, these non parsable sentences can be utilized in additional examination to improve the hindi Parser. There are many parsing calculations accessible. We picked Arc-Eager for checking the parsability of the sentences.

Arc eager -

The arrangement nearly stays as before, yet the lines beginning with 'H' which contained data about that piece, presently likewise contains data about the chunk it is connected to , followed by the checking of the top of that relationship among the two chunk. L implies that the left one is the head, R implies that the correct one is the head. This was followed by the reliance connection between the two pieces as k1/k2 … . This data is put away for each piece isolated by a semicolon ';'.

The word that is the top of the sentence is appeared with the relationship of 'ROOT' with a variable ROOT that is embedded in the stack in bend energetic as we realize that the head expression of the sentence is the ward of that spurious variable in curve excited. The chunks having 'L' relationship will prompt the arrangement of Right circular segment in bend energetic. Also, The lumps having 'R' relationship will prompt the arrangement of Left bend in curve anxious.

Unknown dependencies

For preparing our model, we needed the rundown of a wide range of conditions in our sentences. We had the rundown of all known conditions i.e the pair of words in the sentence that are identified with one another and furthermore the head word among them was set apart by the L

or R. We additionally need to make a few sets of words that are not identified with one another in the sentence to prepare our model consistently so it can likewise foresee the pair of words which are not related. These obscure conditions were made by taking not many of the inconsequential sets from the sentence by our calculation. The head position was set apart by 'U' and their reliance connection was set apart as 'Invalid'.

Format of sentence now -

H यहाँ यहाँ NP PRP NP nmod NP2 ; H दूरी दूरी NP NN NP2 jjmod JJP ; R ; nmod

H दूरी दूरी NP NN NP2 jjmod JJP ; H स्थित स्थित JJP JJ JJP nmod NP3 ; R ; jjmod

H स्थित स्थित JJP JJ JJP nmod NP3 ; H ताल ताल NP NNP NP3 k1 VGF ; R ; nmod

H ताल ताल NP NNP NP3 k1 VGF ; H है है VGF VM VGF NULL ROOT ; R ; k1

H अपने अपना NP PRP NP4 r6 NP5 ; H जल जल NP NN NP5 ccof CCP ; R ; r6

H जल जल NP NN NP5 ccof CCP ; H और और CCP CC CCP rt VGF ; R ; ccof

H और और CCP CC CCP rt VGF ; H है है VGF VM VGF NULL ROOT ; R ; rt

H उसमें वह NP PRP NP6 k7 VGNF ; H उतराते उतरा VGNF VM VGNF nmod__k1inv NP7 ; R ; k7

H उतराते उतरा VGNF VM VGNF nmod__k1inv NP7 ; H हिमखंडों हिमखंड NP NN NP7 r6 NP8 ; R ; nmod__k1inv

Models:

We have used a multi-class linear svm classifier throughout. We have used feature templates as input variables and predict the edge relationships - Left, Right, Unknown/Undefined & Labels

Features fit to the model are combinations of:

- Head words i, j
- Part of Speech tag of head words i, j
- Chunk Tags of chunks

i, j - Chunks passed to the classifier.

Each feature is converted to a one hot vector. All one hot vectors are further joined into a long 1d array. This array is passed to the model in a sparse representation.

Labels to be predicted:

- Edge Dependency Direction - L, R, U
- Edge label: Ex: nmod, jjmod

Total no. of models = 14 (7 + 7 for each label set)

Results - Numeric(Ref to next section for analysis):

Format : Feature - Label , Results (Test set)

Word - L/R/U

	precision	recall	f1-score	support
L	0.75	0.71	0.73	4174
R	0.66	0.77	0.71	10448
U	0.83	0.78	0.80	21561
accuracy			0.77	36183
macro avg	0.75	0.75	0.75	36183
weighted avg	0.77	0.77	0.77	36183

POS - L/R/U

	precision	recall	f1-score	support
L	0.60	0.31	0.41	4174
R	0.67	0.69	0.68	10448
U	0.75	0.81	0.78	21561
accuracy			0.72	36183
macro avg	0.68	0.61	0.63	36183
weighted avg	0.71	0.72	0.71	36183

Chunk Tag - L/R/U

	precision	recall	f1-score	support
L	0.71	0.83	0.76	4174
R	0.67	0.72	0.69	10448
U	0.84	0.78	0.81	21561
accuracy			0.77	36183
macro avg	0.74	0.78	0.75	36183
weighted avg	0.77	0.77	0.77	36183

POS + Chunk - L/R/U

	precision	recall	f1-score	support
L	0.71	0.83	0.76	4174
R	0.67	0.70	0.68	10448
U	0.83	0.78	0.80	21561
accuracy			0.76	36183
macro avg	0.73	0.77	0.75	36183
weighted avg	0.77	0.76	0.76	36183

Word + Chunk - L/R/U

	precision	recall	f1-score	support
L	0.74	0.79	0.77	4174
R	0.67	0.79	0.72	10448
U	0.86	0.77	0.81	21561
accuracy			0.78	36183
macro avg	0.75	0.78	0.77	36183
weighted avg	0.79	0.78	0.78	36183

Word + POS - L/R/U

	precision	recall	f1-score	support
L	0.75	0.72	0.73	4174
R	0.66	0.79	0.72	10448
U	0.85	0.77	0.81	21561
accuracy			0.77	36183
macro avg	0.75	0.76	0.76	36183
weighted avg	0.78	0.77	0.78	36183

Word + POS + Chunk - L/R/U

	precision	recall	f1-score	support
L	0.74	0.79	0.77	4174
R	0.66	0.79	0.72	10448
U	0.86	0.77	0.81	21561
accuracy			0.78	36183
macro avg	0.75	0.78	0.77	36183
weighted avg	0.79	0.78	0.78	36183

Note : Only abridged results here. Detailed Results in Notebook in this form:

	precision	recall	f1-score	support
K2U	0.00	0.00	0.00	1
ROOT	1.00	1.00	1.00	11145
UNDEF	0.00	0.00	0.00	1
adv	0.73	0.54	0.62	1325
ccof	0.81	0.90	0.85	12223
enm	0.00	0.00	0.00	2
fragof	0.50	0.22	0.31	9
jjmod	0.91	0.95	0.93	716
jk1	0.00	0.00	0.00	10
k1	0.31	0.88	0.46	13000
k1s	0.00	0.00	0.00	1487
k1u	0.00	0.00	0.00	46
k2	0.40	0.54	0.46	10234
k2g	0.00	0.00	0.00	31
k2p	0.00	0.00	0.00	387
k2s	0.00	0.00	0.00	296
k2u	0.00	0.00	0.00	34
k3	0.00	0.00	0.00	374
k3u	0.00	0.00	0.00	1
k4	0.00	0.00	0.00	1595
k4a	0.00	0.00	0.00	216
k4u	0.00	0.00	0.00	1
k5	0.00	0.00	0.00	918
k7	0.00	0.00	0.00	6356
k7a	0.00	0.00	0.00	542
k7p	0.00	0.00	0.00	3537
k7pu	0.00	0.00	0.00	6
k7t	0.00	0.00	0.00	4704
k7tu	0.00	0.00	0.00	3
k7u	0.00	0.00	0.00	8
mk1	0.00	0.00	0.00	1
nmod	0.99	0.20	0.33	3817
nmod__emph	0.00	0.00	0.00	23
nmod__k1inv	0.72	0.96	0.82	902
nmod__k2inv	0.00	0.00	0.00	258
nmod__pofinv	0.00	0.00	0.00	10
nmod__relc	0.74	0.92	0.82	238
pk1	0.00	0.00	0.00	7
pof	0.75	0.35	0.48	8430
r6	0.70	0.95	0.81	12126
r6-k1	0.00	0.00	0.00	346
r6-k2	0.00	0.00	0.00	1756
r6v	0.00	0.00	0.00	15
rad	0.00	0.00	0.00	5
ras-NEG	0.00	0.00	0.00	49
ras-k1	0.00	0.00	0.00	528
ras-k2	0.00	0.00	0.00	137
ras-k4	0.00	0.00	0.00	4
ras-k7	0.00	0.00	0.00	7
ras-k7p	0.00	0.00	0.00	5
ras-pof	0.00	0.00	0.00	6
ras-r6	0.00	0.00	0.00	2
ras-r6-k2	0.00	0.00	0.00	3
ras-rt	0.00	0.00	0.00	1
ras__k7	0.00	0.00	0.00	1
rbmod	0.88	0.97	0.92	30
rbmod__relc	0.65	1.00	0.79	24
rd	0.00	0.00	0.00	264
rh	0.00	0.00	0.00	856
rs	0.00	0.00	0.00	171
rsp	0.00	0.00	0.00	116
rsym	1.00	1.00	1.00	11078
rt	0.33	0.44	0.38	1615
rtu	0.00	0.00	0.00	1
sent-adv	0.00	0.00	0.00	531
vmod	0.97	0.61	0.75	2501
vmod__adv	0.00	0.00	0.00	1
accuracy			0.61	115073
macro avg	0.18	0.19	0.17	115073
weighted avg	0.56	0.61	0.55	115073

Word - Edge Relationship Labels

accuracy			0.75	14622
macro avg	0.39	0.33	0.35	14622
weighted avg	0.73	0.75	0.73	14622

POS - Edge Relationship Labels

accuracy			0.61	14622
macro avg	0.22	0.21	0.20	14622
weighted avg	0.59	0.61	0.56	14622

Chunk Tag - Edge Relationship Labels

accuracy			0.63	14622
macro avg	0.22	0.23	0.21	14622
weighted avg	0.58	0.63	0.57	14622

POS + Chunk -Edge Relationship Labels

accuracy			0.64	14622
macro avg	0.25	0.24	0.23	14622
weighted avg	0.61	0.64	0.59	14622

Word + Chunk - Edge Relationship Labels

accuracy			0.77	14622
macro avg	0.41	0.36	0.38	14622
weighted avg	0.76	0.77	0.76	14622

Word + POS -Edge Relationship Labels

accuracy			0.77	14622
macro avg	0.41	0.36	0.37	14622
weighted avg	0.76	0.77	0.76	14622

Word + POS + Chunk - Edge Relationship Labels

accuracy			0.77	14622
macro avg	0.41	0.37	0.38	14622
weighted avg	0.76	0.77	0.76	14622

Conclusions and inferences :

- Overall accuracy of models is around 77% for both L/R/U and Edge Relationship Labels.\
- Data can be interpreted as an Ordered Set of Chunk Features.
- What does the overall 77% accuracy mean?
- A linear svm classifier is being used. It can be said that the ordered pairs are quite distinct in terms of the regions they lie in.
- Consider using less features: Only words, chunks or pos or 2 of those instead of 3. Accuracy difference is miniscule. Upon further analysis we see slight differences in trend in f1 scores
- POS tag -> L/R/U does terrible in Precision/Recall/F1 score. The accuracy seems to be high - 72% due to distribution of samples pulling it up. The data being high dimensional, and as previously argued decently linearly separable, it is fitting very strongly to the probability distribution of labels. I believe if the test set had different ratios of L/R/U, this would perform far worse. Also, this predicts mostly unknown, so it is meaningless in practice.

POS - L/R/U

	precision	recall	f1-score	support
L	0.60	0.31	0.41	4174
R	0.67	0.69	0.68	10448
U	0.75	0.81	0.78	21561
accuracy			0.72	36183
macro avg	0.68	0.61	0.63	36183
weighted avg	0.71	0.72	0.71	36183

- Hence other models have close to best performance observed in Word + Pos + Chunk for both sets of labels. From this we infer -
    - More than POS is necessary to make an informed robust decision.

    - Combinations of features are not being fully utilized. Regardless of the combination we seem to be hard capped at 77~78%.
- A learnable weighing mechanism might be a good solution to this.
- Edge relation label predictions in some cases return 86% on train set, but still 77% range for test set.
- Same inferences as POS tag - L/R/U are valid.
- Some classes such as ROOT have F1 score 1. (ROOT is a special case).
- Some have 0 F1 score.
- These are due to the huge difference in the amount of instances of each class.
- Accuracy is indeed very good, however, this is not a metric. This simple model is insufficient to achieve a robust dependency parse.
- Reference 14 provides a baseline that our process is correct, as accuracy scores match.

References

(1) https://nlp.stanford.edu/software/nndep.html#:~:text=A%20dependency%20parser%20analyzes%20the,parse%20of%20a%20short%20sentence.&text=A%20Fast%20and%20Accurate%20Dependency%20Parser%20Using%20Neural%20Networks.

(2) https://cl.lingfil.uu.se/~nivre/docs/eacl3.pdf

(3) https://universaldependencies.org/treebanks/hi_hdtb/index.html#:~:text=The%20Hindi%20Universal%20Dependency%20Treebank,developed%20at%20IIIT%2DH%20India.

(4) http://verbs.colorado.edu/hindiurdu/guidelines_docs/ssf-guide.pdf

(5) http://ltrc.iiit.ac.in/treebank_H2014/

(6) Neural dependency parsing: https://www.aclweb.org/anthology/D14-1082.pdf

(7) Parsing with LSTM representations: https://www.aclweb.org/anthology/Q16-1023.pdf

(8) code-mixed parsing: https://www.aclweb.org/anthology/N18-1090/

(9) analysis of dependency parsing: https://www.aclweb.org/anthology/D18-1278.pdf

(10) Feature template - https://www.aclweb.org/anthology/P11-2033.pdf

(11) contextual representations - https://arxiv.org/pdf/1909.08744.pdf

(12) https://www.aclweb.org/anthology/W12-3410.pdf use of svms for dependency parsing in Tamil using model agreement features.

(13) https://web.stanford.edu/~jurafsky/slp3/14.pdf. Speech and Language Processing. Daniel Jurafsky & James H. Martin. Copyright © 2020. All rights reserved. Draft of December 30, 2020. Chapter - 14

(14) http://web2py.iiit.ac.in/research_centres/publications/download/inproceedings.pdf.b956a3308fcf616c.436c6173736966669657220456e73656d626c6520617070726f6163682e2e2e2853696c7061204b616e6e6567616e74692e706466.pdf