Measuring Sequence Iconicity

Abhishek Sarangi

A Dissertation

Presented to the University of Dublin, Trinity College in

partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Data Science)

Supervisor: Prof. Tim Fernando

September 7, 2020

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Abhishek Sarangi

September 7,2020

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Abhishek Sarangi

September 7, 2020

Acknowledgments

I would like to thank Professor Tim Fernando for his consistent and relentless guidance during the course of this project. I would also thank Computer Science and Statistics Department of Trinity College Dublin, Ireland for providing the infrastructure and support for the completion of this research project.

I would also like to thank my second reader Professor Owen Conlan for his valuable comments and feedbacks on this project.

I would also thank my family and friends for their continuous support and motivation in this intense pandemic time of the year to complete this project.

Abhishek Sarangi

University of Dublin, Trinity College
September,2020

Measuring Sequence Iconicity

Abhishek Sarangi, Master of Science in Computer Science

University of Dublin, Trinity College, 2020

Supervisor: Prof. Tim Fernando

Abstract

This paper describes the temporal relations of events. Events can be described in the same order as they occur but sometimes they may be described otherwise. TimeML is a way to describe a given text by means of annotation. It is an excellent specification language for annotating event and time related or temporal expression in a text consisting of natural language. Studying the iconicity of temporal relations is important to know what prompts a speaker or a writer to write one way or the other. The iconicity principle states that the positioning of subordinate and main clauses in a text follow the same order as they have occurred. Besides iconicity principle factors such as length, complexity and pragmatic import may also affect the positioning of adverbial clauses. There are thirteen Allen Relations which can be used to annotate a given text. There have been several studies on how to annotate a given text to know the temporal relations between events. In this paper we attempt to use the machine learning algorithms to predict the iconicity in terms of 'BEFORE' and 'AFTER' when a certain combination of annotations is given.

Summary

Iconicity principle states that the positioning of subordinate and main clauses follow the same order in a text as they have occurred in reality. There are several previous works where an attempt has been made to represent time and event information in a text. One of the best forms of doing that was using TimeML. TimeML is a way of annotating a given text so that the final annotated document looks like a marked up language just like an HTML file. The tags give the information of temporal events. They also give information about events. The tags also establish a relation between time and another time or between time and event or between event and event. There are thirteen Allen relations which describe how two events may be related to each other. TimeML uses Allen relations to annotate a text and establishes relations between temporal events. The basic ones are if two events are related by 'BEFORE' or 'AFTER'. In other words, two events may be related to each other by 'BEFORE' which mean one event may have occurred before another event. They can also be related by 'AFTER' which means that one event may have occurred after another event. In this paper such tags and relations have been studied in detail and a relation has been drawn out of such information. All the details of the annotation were collected in a csv file from a time bank corpus of news data and different machine learning algorithms and deep learning have been applied to them. The results have been studied in terms of whether each of the thirteen Allen relations could be described by either 'BEFORE' or 'AFTER'. This gives an idea of measuring the iconicity of a text and also annotating the text better to improve the accuracy of the measurement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Temporal relations between events and times are important to understand the semantic of a text. Recently, NLP has been used to understand this relation and the Data Science community is really interested in it. Establishing a relation between temporal events is likely to give us a meaning of the text. It is a part of understanding the natural language. It is helpful in many applications such as construction of story timeline, question answering, summarizing a text, extracting information from a given text and others. It has attracted attention in recent years due to its applications in understanding the natural language.

The research is primarily based on establishing a temporal relation between a pair of events, an event and an expression of time or between pairs of expressions of time. There have been many attempts to annotate a document to establish a relation between these pairs but the most successful is by using TimeML. TimeML is a markup language which uses Allen relations to establish a relation between the pairs. There are thirteen Allen relations which can be established between pairs. A document when annotated looks exactly like an HTML file except for the difference that it links pairs of events, event and time and times. The thirteen Allen relations which are used by TimeML are ‘before’, ’after’, ’immediately before’, ‘immediately after’, ’begins’, ‘begun-by’, ‘ends’, ‘ended-by’, ‘includes’, ‘included by’, ‘simultaneous’, ‘during’, ‘identical’.

TimeML is also used to study iconicity principle which states that the main and subordinate clauses follow the same linear order in a text as they have occurred. In this paper we use a time bank corpus data to analyze the temporal relations by using different machine learning and deep

learning algorithms. The time bank corpus uses the news of different channels like ABC, CNN and APW. The contexts of the news are different and they are not of the same genre. It has also been argued that the length of the text can also affect the positioning of the main and adverbial clauses. However we are more concerned with the annotation that gives a semantic aspect to the text.

TimeML which was developed under AQUAINT program is a rich specification language which was an effort to access the information from a text from the content than keywords. The question answering systems was its main focus. There are four major aspects to a TimeML annotation. These are described as follows.

a. Time stamping – Time stamping by identifiying events is necessary to identify which event occurred at what time. Then these times can be referred to in other places of the document.

b. Ordering – Ordering of events, event and time and times is important to know which order the events have occurred. This ordering is done with respect to each other. Events and times follow Allen relations and hence they have to be described by one such relation.

c. Clarity or reasoning – Clarity or reasoning of unclear or limited data temporal expressions. Facts such as ‘last week’ or ‘two weeks before’ are not strong temporal expressions to denote a particular point of time.

d. Clarity on the lasting of events - Clarity on the lasting of events is important to know which event lasted for what amount of time. This fact is necessary to see the outcome of an event and also see at what point of time the outcome came and how long it lasted.

   Let us take an example of a sentence. The sentence is “John pushed Mary. Mary fell down.” This sentence can be spoken in many different ways. One could be “John pushed Mary so she fell down.” Or “Mary fell down because John pushed her.” Any way we say the meaning or the semantic of the statement is the same but the ordering of the events are different in

different sentences. For example “John pushed Mary so she fell down” follows the iconicity as the cause comes before the effect. The sentence “Mary fell down because John pushed her.”
does not follow iconicity as the cause comes later than the effect in the sentence. This paper attempts at finding out such annotated texts and understand them to predict if a certain sentence follows iconicity or not. In this paper however it is more like finding which sentence’s Allen relations are like “BEFORE” or “AFTER”.

Let us take an example from an annotated text. The annotated text is taken from a time bank corpus of news data.

<s>
On the other hand, it's
<EVENT eid="e1" class="OCCURRENCE">turning</EVENT>
out to be another very
<EVENT eid="e369" class="STATE">bad</EVENT>
financial
<TIMEX3 tid="t83" type="DURATION" value="P1W" temporalFunction="false" functionInDocument="NONE">week</TIMEX3>
for
<ENAMEX TYPE="LOCATION">Asia</ENAMEX>
.
</s>

This is an excerpt from a document. The tag <s> symbolizes the beginning of a sentence and the tag </s> symbolizes the end of the sentence. The verbs and the related events and time expressions have been marked up by <EVENT> tag and <TIMEX3> tag respectively. The event e1 may be related to the event e369 by one of the Allen relations. It can also be related to tid t83. The relations are not specific to only one sentence. The event e1 for example may be related to some other event in some other part of the document. The relations are described by TLINK tags. One such example is as follows.

<TLINK lid="l1" relType="BEFORE" eventInstanceID="ei377" relatedToEventInstance="ei378"/>

As we can see the TLINK tag has an id l1, the “relType” attribute describes the Allen relation between the instances eventInstanceID ei377 and the eventInstanceID ei378. The eventInstanceIDs are MAKEINSTANCE ids which give further information about a particular event such as tense, aspect, polarity etc. One such example of MAKEINSTANCE id is as follows.

<MAKEINSTANCE eventID="e4" eiid="ei378" tense="PRESENT" aspect="PROGRESSIVE" polarity="NEG" pos="VERB"/>

As we can see the MAKEINSTANCE has eventID “e4”. So this event id can be seen in some EVENT tag. MAKEINSTANCE has its own id called eiid which is used in TLINK tags. There are other attributes such as tense, aspect, polarity and pos. These give extra information about the event.

TimeML gives extra features that help in separating the representation of events and their orderings. During 1990s there was a significant development in the field of computational linguistics and in that time a standard annotation scheme was to be developed to annotate the documents which had temporal expressions in them. And around that Translingual Information Detection Extraction and Summarization (TIDES) helped in making an annotation scheme. TimeML was made in 2002 by a group of researchers including Professor James Pustejovysky [3]. The idea was conceptualized in a workshop called TERQAS which stands for Time and Event Recognition for Question Answering Systems. It was basically to address the problem of question answering systems which could answer based on the content.

1.2 TimeML tags

TimeML has many tags to annotate a document. Some of them are listed below and an explanation of them along their attributes has been given.

<EVENT> tag

Events can be anything that happens. They occur and can take place at a point of time or last for a duration of time. There are several attributes of an event. They are explained below.

a. EVENT ID number (eid) – This id number is used uniquely to identify an event. An annotation tool automatically assigns this id to any event when a string is associated with an event.

b. CLASS – This is a mandatory attribute and every event has it. It tells what type of event that is.
A verb in one case may describe an event of one class but may describe another event of another class. The different types of classes are as follows.

REPORTING – This is just like the name goes as it reports, says or narrates an event. For example, John said that Mary fell down.

PERCEPTION – As the name suggests this deals with physically witnessing another event. Typical verbs which express this are see, watch, view, hear etc. For example, John said that he saw Mary falling down.

ASPECTUAL – This feature describes the different aspects or facets of an event. There are situations where we indicate the beginning, restarting, termination, culmination and continuation of events. For example, John began to do his homework after four hours.

I_ACTION - These are arguments which do not have necessarily happened when the I_ACTION events take place. They give an idea about something when its relation with I_ACTION is given. Some common words describing such are attempt, investigate, delay etc.

I_STATE – This describes the possibility of something. For example, John now feels that he must do his homework before ten o'clock.

STATES - These are the situations where something is true. In a document only those situations are marked up which change over the course of the document. For example, 'The earth in round' cannot be marked up as it is always true. However, 'Only two people have 'come' so far' can be marked as the state that only two have come so far can change. Often state is described in TIMEX3 tag.

OCCURRENCE – This class describes some additional events that happen. For example, Fifty planes landed in the country.

<TIMEX3> tag

TIMEX3 tag is used to mark up a time related to an event. Let us take the example below.

I have been exercising for two hours daily since last month. <TIMEX3 tid="t1">two hours</TIMEX3from<TIMEX3 tid="t2">last month</TIMEX3>

These two time events can be related by a TLINK. The attributes of TIMEX3 can be used to describe particular times such as dates or times and also uncertain times such as 'last week'. The important attributes of the TIMEX3 tag are as follows.

ID number – The id number is represented by tid. Every TIMEX3 tag has a unique id number and this is a mandatory attribute. The annotation tool automatically assigns it.

Type - Type determines the type of time it is indicating. It can be DATE, TIME, DURATION or a SET.

DATE - This describes a calendar time. The italicized words below form a part of DATE.

He left on *Wednesday, Jan 1, 2020*.

TIME – This attribute refers to the time of a day.

He left yesterday at 8 am.

DURATION – This attribute describes duration.

He stayed here for two hours.

SET – This attribute represents a set of times or the frequency with which something happens.

He goes there *twice a week*.

The other not so important attributes of TIMEX3 tags are below. They convey some information on how the above date, time, duration and set occur. They are 'Value','mod','temporalFunction','anchorTimeID','valueFromFunction','functionInDocument','beginPoint and endpoint', 'quant and freq'.

<MAKEINSTANCE> tag

The modality, tense, polarity and part of speech are determined by the MAKEINSTANCE tag. There may be instances where a verb represents two or more different events. In such cases MAKEINSTANCE tag can represent the cardinality of the occurrence. Example like John plays football on Monday and Thursday. In this the event 'plays' happens on Monday and Tuesday. Therefore either two MAKEINSTANCE tags can be made to represent the events or the cardinality can be mentioned in a single tag. The following example shows how a MAKEINSTANCE tag would look like if it were written for the example above.

<MAKEINSTANCE eiid="ei1" eventID ="e1" tense="PRESENT" aspect="NONE" pos="VERB"/>

<MAKEINSTANCE eiid="ei2" eventID ="e1" tense=" PRESENT" aspect="NONE" pos="VERB"/>

If only one MAKEINSTANCE tag were written then it would look as follows.

<MAKEINSTANCE eiid="ei1" eventID="e1" tense=" PRESENT" aspect="NONE" pos="VERB" cardinality="2"/>

MAKEINSTANCE has a unique ID number called eiid. It also has an attribute eventID which represents the event it is describing. There are other attributes like tense which says the tense of the event. The aspect attribute says whether it is simple, progressive, perfective or perfective progressive. The pos attribute tells whether the event marked is a noun, verb or adjective.

1.3 Uses

There are several uses of temporal annotations. It has attracted the natural language processing communities and computational linguistics. The events can be related to each other and may be times can be related to each other and may be an event and a time can be related to each other. It is a difficult task to figure out all the temporal relations between the events in a document. However, schemes like Allen's interval algebra do provide some relations that can be applied to events in a document. It is helpful in information extraction, text mining, analyzing social media and digital history. It is helpful in finding if the facts presented in a text are true or not.

Chapter 2

Related Works

In [8] they use ISO 8601 to represent time values. They take into account indexicality, fuzziness of boundaries, context-dependence, granularity and ambiguity and extended the ISO standard in many different ways. Work of Setzer and Gaizauskas represents time values and fine grained time and event-time temporal expressions. Another work is from Stanford Knowledge Systems Lab which places time points on a number line and places the events on them to represent which happened before or after another event. It also takes care about the granularity of time so that proper time points can be marked on the number line and accordingly the events can be marked on it.

The other effort which is known as TERQAS developed an annotation scheme which helps in ordering the events on a time line. It focuses on then syntactic aspects of a text document to describe the order of events that as described in natural language. Much work can be found in clinical domain. In [1] the THYME guidelines were developed to annotate a corpus of clinical notes. In this, linguistically- derived and inferentially-derived temporal relations or orderings have been distinguished. It is available to the community.

Another effort was done by Reichenbach in 1947 [2] when he proposed a three-point time system to describe tenses of events. He focused more on verbs to make three points. Those three points are speech, event and reference. The speech time refers to the time when the verb expressed is spoken or written. The event time is the time in which the event actually occurs. The position of this point relative to other verbs describes the temporal order of events. The third point is the reference point from which events are viewed.

Clinical narratives from Informatics for Integrating Biology and the Bedside which is also known as i2b2 have been annotated for developing temporal reasoning systems. The corpus is available for the community. The process followed for the development of the i2b2 corpus is well described (Sun W, Rumshisky A, Uzuner O. Annotating temporal information in clinical narratives. J Biomed Inform. 2013;46 Suppl(0):S5-S12. doi:10.1016/j.jbi.2013.07.004).

A corpus based analysis of spoken and written English and identifying the effect on iconicity based on the positioning of adverbial and main clause in a sentence has been studied in [4].Logistic regression has been used to study the positioning of adverbial and main clauses and its effect on iconicity.

Chapter 3

Iconicity

Iconicity is the study of sequential order of events. It means it studies whether the sequential order of events is reflected or mirrored in the speech or text or not. For example, if we take a sentence ‘John pushed Mary. Mary fell down’. The statement contains two events. The first one is John pushing Mary. Let us call it e1. The second is Mary falling down. Let us call it e2. The meaning or the semantic of the statement is clear and it is that John pushed Mary which is the action or the cause and Mary fell down which is the effect. It is also clear that pushing is followed by falling. So, the sentence ‘John pushed Mary. Mary fell down.’ follows iconicity because it is in the same order as the reality. On the other hand, if the text was ‘Mary fell down because John pushed her’ then it would have not followed iconicity for the fact that the effect has been stated first and the cause after that. It can be used to study how and what influences a speaker to speak one way than the other. This is also specific to languages. The percentage of iconicity followed and not followed may not be very same in all the languages. In some languages there is fair distribution of iconicity followed and not followed while in some other languages people might prefer to not follow the iconicity as the language might have evolved like that. Whatever may be the case it needs a deep analysis of all the languages and not only the spoken form of that language but also the written form of that language.

Chapter 4

Allen Relations

James F Allen in 1983 gave a set of thirteen relations which can exist between a pair of temporal events. It is extremely useful in making the TimeML tags and in TLINK tags they are used to relate two events, an event and a time or two times. If we consider the one event as X and another event as Y then the table following shows the relations can be established between them.

The table below gives a clear picture of Allen Relations, their illustrations and the TimeML tags. On the semantic side these thirteen Allen Relations exist but on the textual side they can be classified as either BEFORE or AFTER as per the positions of the events in the document. This reduction from thirteen to two Allen Relations is important to understand the behavior of all the Allen relations. This can definitely help us in understanding what might have prompted the writer to write one way or the other.

S no	Allen's Relation	Illustration	TimeML Relation
1	X<Y, Y>X	X---------- Y----------	X BEFORE Y, Y AFTER X
2	X m Y, Y mi X	X---------- Y----------	X IBEFORE Y, Y IAFTER X

3	X o Y, Y oi X	X---------- Y----------	X overlaps with Y
4	X s Y, Y si X	X---------- Y----------	X BEGINS Y, Y BEGUN_BY X
5	X d Y , Y di X	X---------- Y----------	X DURING Y, Y DURING_INV X
6		X---------- Y----------	X INCLUDES Y, Y IS_INCLUDED X
7	X f Y, Y fi X	X---------- Y----------	X ENDS Y, Y ENDED_BY X
8	X=Y	X---------- Y----------	X SIMULATNEOUS Y

Table 4.1: Illustration of Allen Relations

The first row marked as 1 shows two relations BEFORE and AFTER. X is an event that happened before Y. X started at some point in time and ended. Y followed the ending of X and started after sometime and then ended. AFTER and BEFORE are inverse of each other so it is convenient to say X happened before Y or Y happened after X. The second row marked as 2 shows two relations IBEFORE and IAFTER which simply stand for immediately before and immediately after. They are inverse of each other. It means X started at some point in time and ended. Y started that very moment at which X ended and then it also ended. The row 3 shows some overlapping of two events. X starts at some point in time and while it is going on another event Y starts somewhere in between and then X ends after which Y ends after sometime. The fourth row is X and Y start together but then X ends before Y ends. This is represented by X BEGINS Y or by its inverse Y BEGUN_BY X. The fifth row is a 'during' relation between two events. Here an event Y starts and then somewhere in between the duration of Y, X starts and before Y completes X ends. So it means one event sees a different event happening within its duration. It is represented by X DURING Y. Its inverse is Y DURING_INV X. Another way that it can be represented is by INCLUDES tag. What if X has the longer duration and Y has the shorter duration but Y comes within the duration of X? Then the concept is the same except for it can be said that X INCLUDES Y or Y IS_INCLUDED X. Another case when two events end together at the same point in time but one had started somewhere in between the duration of the second event is the case of row number 7. The illustration shows that X started somewhere in between the duration of Y but the both ended together. This is represented in TimeML tag as ENDS and ENDED_BY tag which is the inverse of ENDS tag. So, in this case it has been represented as X ENDS Y and its inverse is Y ENDED_BY X. The last row 8 represents the relation between two events that can exist when both of them occur at the same time. It means they start at the same time and continue for the same amount of time and end together at the same point of time. It is represented in TimeML tag by SIMULTANEOUS tag. SIMULTANEOUS tag has no inverse as both events go on together. Based on these Allen relations a document is annotated using the TimeML tags.

Chapter 5

Methods

5.1 Collection of data

The collection of data was a very crucial part of this project. Unlike other research projects where data is abundant this research project required a dataset with enough data to analyze and build a model on it. Data collection is an important task for any research project. For this project the time bank corpus which was annotated with TimeML tags was used. A total of 183 files containing news bulletin are the corpus of dataset. All the files are annotated with the TimeML scheme. The news contains different topics. However, in most of the news bulletins in a single file only particular news has been focused. For example, if a file or news bulletin talks about tensions in Iraq with America then the whole news bulletin talks about the same issue. So, there is no variation in the news in a single file but there is a variation of news in different files. The data looks good in terms of annotations because all the aspects of TimeML have been covered in the annotations. This posed a problem because to study the data through machine learning algorithms they should be in a different format. The usual convenient format is a csv file with rows and columns. The rows represent the data and the columns have the attributes or the properties of the data.The breakdown of the contents of a file are the starting of a file which contains the timestamp at which the news was made followed by the description of the events and important times which are then explained by other tags such as MAKEINSTANCE tags. After that the temporal relation tags called TLINKs follow which relate two events or times or an event and a time by one of the Allen relations. Reading data from these files was convenient but not compatible with the code which applies machine learning algorithms. The files had to be parsed to extract the information they had to form a csv file or any other format to be conveniently read by the code. Following is an example of the contents of a file. It contains a small part of a single file.

<TimeML

xsi:noNamespaceSchemaLocation="http://timeml.org/timeMLdocs/TimeML_1.2.1.xsd">

APW19980213.1380 NEWS STORY

<TIMEX3 tid="t30" type="TIME" value="1998-02-13T15:44:00" temporalFunction="false" functionInDocument="CREATION_TIME">02/13/1998 15:44:00

</TIMEX3>

w2227 Cx1f wstm- r i Cx13 Cx11 BC-PuertoRico-HostageKil 02-13 0243 BC-Puerto Rico-Hostage Killed,0246 Police

<EVENT eid="e41" class="OCCURRENCE">discover</EVENT>

<EVENT eid="e40" class="OCCURRENCE">dismembered</EVENT>

body of man

<EVENT eid="e39" class="OCCURRENCE">kidnapped</EVENT>

Wednesday UR By HILARIO DE LEON QC UR Associated Press Writer QC CAGUAS, Puerto Rico (AP) _ Kidnappers

<EVENT eid="e1" class="ASPECTUAL">kept</EVENT>

their

<EVENT eid="e42" class="I_ACTION">promise</EVENT>

to

<EVENT eid="e2" class="OCCURRENCE">kill</EVENT>

a store owner they

<EVENT eid="e3" class="OCCURRENCE">took</EVENT>

<EVENT eid="e43" class="OCCURRENCE">hostage</EVENT>

and police

<EVENT eid="e4" class="OCCURRENCE">found</EVENT>

the man's

<EVENT eid="e5" class="OCCURRENCE">dismembered</EVENT>

and

<EVENT eid="e44" class="OCCURRENCE">decapitated</EVENT>

Body

<TIMEX3 tid="t33" type="DATE" value="1998-02-13" temporalFunction="true" functionInDocument="NONE" anchorTimeID="t30">Friday</TIMEX3>
<EVENT eid="e6" class="OCCURRENCE">wrapped</EVENT>
in plastic garbage bags.

The part above describes the events that happened. For example, the news is “Police discover dismembered body of a man kidnapped Wednesday UR By HILARIO DE LEON QC UR Associated Press Writer QC CAGUAS, Puerto Rico (AP)_Kidnappers kept their promise to kill a store owner they took hostage and police found the man’s dismembered and decapitated body in plastic garbage bags.”

<MAKEINSTANCE eventID="e42" eiid="ei306" tense="NONE" aspect="NONE" polarity="POS" pos="NOUN"/>
<MAKEINSTANCE eventID="e2" eiid="ei307" tense="INFINITIVE" aspect="NONE" polarity="POS" pos="VERB"/>

The MAKEINSTANCE tag assigns an id eiid to itself and also takes an event id and gives some information about the event’s tense, aspect, polarity, pos etc.

<TLINK lid="l51" relType="BEFORE" eventInstanceID="ei306" relatedToEventInstance="ei307"/>
The TLINK then relates the two events by an Allen Relation which is done by a TimeML tag. In the above tag it is done by a BEFORE tag and it says that event ei306 happens before ei307.

The above description of data needed to be carved out of the files and for this a python code was written to parse the file. The code would read all the files in its working directory with a .tml extension and get all the TLINK tags of them. It would also pick the attributes related to them. For this research attributes lid, signalID, tid1, temporalFunction, functionInDocument, event, tense, aspect, polarity, pos and class have been picked for building a model. All the 183 files were read and a single csv file with 4604 rows and 21 columns was made. The

dataset contains both the temporal events and their attributes. In the csv file they are represented as temporalFunction1, functionInDocument1, temporalFunction2, functionInDocument2 etc. Of all the columns 20 columns are features upon which the 21st column is decided. The last column has been filled with either 0 or 1. 0 represents a 'before' relation and 1 represents an 'after' relation. This means any Allen relation has been classified as a BEFORE or AFTER relation. So, the entire annotated document has been depicted as a table of rows and columns. This is how any number of files can be parsed and a huge dataset can be made which is easier to study.

Considering the fact that in future the annotations may change and the code can be useless then it is only a matter of changing the required annotation tags in the python code to parse the newly annotated files. The code can also be made dynamic to accommodate any changes so that no hard coding of tags is done in the code and some dynamic substitution of tags can be done to parse the new documents to make a csv file or the dataset that can be used for any machine learning algorithm. So it is very important first to get all the information related to the events out of the annotated document and prepare a dataset out of it as a human annotator would only make an XML based document and would not make it for only machine learning algorithms but for general use. It is up to the researchers how to use that data and do some research on it.

5.2 Preparation of data

After the collection of data from the xml based annotated document the dataset has to be studied. This study is important to get an insight of the dataset. The insight can be based on some standard rules and regulations and there may be some personal perspective also. The dataset made by parsing the documents contained 4,604 rows or data and the number of columns or features for each of this data is 21. The important features are lid which describes the TLINK id, signalID which describes the prepositions like on, in, for relations between two sentences, tid1 the time id of the first time encountered in a TLINK tag, temporalFunction1 the temporal function of the first event which explains the vaguely represented time like

‘every Monday’ and not a particular time like ‘Friday at 9 pm’, functionInDocument1 that indicates that a TIMEX3 tag serves as a temporal anchor for other events in the document. The MAKEINSTANCE tag has an attribute ‘tense’ which

shows the tense of the event for which it is related to. There are other attributes aspect1, polarity1, and pos1. The same attributes for the second event or time are suffixed with a 2 like tid2, temporalFunction2, functionInDocument2 etc. There is another attribute which is in EVENT tag and that is the ‘class’ attribute which tells what the event was like. For example, an event can be like ‘Occurrence’,’State’,’Reporting’,’I-Action’,’I-State’,’Aspectual’,and ’Perception’. The csv also has a column called ‘Iconicity’ and determines if a particular relation of events could be classified as BEFORE or AFTER indicated by 0 or 1 respectively in the column.

This is the basic structure of the dataset created. Most of the columns except the ‘Iconicity’ column contain string values in them. This research is based on measuring sequence iconicity. It has been done by applying several machine learning algorithms to the dataset. The main aim is to understand if a given set annotation attributes indicate the iconicity. All the columns had to be converted to numerical values to represent the string values. For this a python code helped in applying the fundamental rules of machine learning data cleaning to encode the values. Encoding is a technique by which we represent any labels or categorical data into integer data or numeric form. This makes the structured data more readable by the machine and hence the algorithms can operate on them better. In this case one hot encoding has been used. In one hot encoding a binary vector representation of the categorical data is used. As in any of the columns here like the tense column the values in them do not have any natural ordering. Therefore, it is good to use one–hot encoding to remove biased or unpredictable prediction. For example, if we look into the tense column then we can find values like ‘SIMPLE’, ‘PRESENT’,’PROGRESSIVE’. These do not have any natural ordering. Therefore, one hot encoding helps in such situations as integer encoding that is encoding them as 1,2 and 3 would make ‘SIMPLE’ superior or may be ‘PROGRESSIVE’ more significant than ‘SIMPLE’ and ‘PRESENT’.

The entire dataset was encoded with one hot encoding to encode the categorical values and this way the dataset now had only numerical values. The other important part was to handle the null values. Several columns had null values in them. They had to be removed or if necessary had to be represented in some way to make the model understand that that particular value did not exist for that particular event or time. In this research the null values were not removed but instead given a value and were encoded to make it understandable to the machine that a particular attribute assumes a particular kind of value when it is an event and other kind of value when it is a time. This way the whole encoding was done. This non-removal of null values was important as there were several null values in some columns for some rows while they had values in them for some other rows. So, the idea to give them a particular value was just to save the removal of rows as the dataset was small and any removal of rows would have reduced the size of the data and impacted the efficiency of the algorithm. Indeed, there can alternatives to this but when the data size is less it is important to keep the data size intact.

5.3 Background of Methods Used

5.3.1 DecisionTreeRegressor

A decision tree regressor uses a series of yes no questions based on the labels in the training data to build a decision tree that if given a set of features can follow the same established set of path in the decision tree to reach an answer. It contains decision nodes where the concerned attribute or feature of the dataset is tested. It may split into two or more branches based on the outcome of the decision node. Continuing this decision tree ends in leaf nodes. A decision tree can handle both categorical and numerical data. The crucial part of a decision tree is the split on the nodes. This decides how good a decision tree is. For classification problems information gain or the entropy is used to decide if a split

has to be done in a node or not. For example, if the information gain is more based on a feature then the node can be split up to two or more branches based on how many classification can be done based on the split. In a regression problem reduction in variance is used to check if a split in a node is required or not. Variance is a measure of how far the values are spread out from its mean. It is usually measured by the MSE or Mean Squared Error of the children nodes. In a children node it is measured by the weighted mean squared error. In a node it means how pure the node is or how homogenous the node is. If a child node after split contains more similar kind of elements in it then definitely the feature used to split the node was significant and it can be decided with a greater certainty that that feature gives a good decision or a number in case of regression. If a node has zero MSE that means it has no error or the predicted values and true target values are the same. The node contains similar elements with no mixture in it. Below is the plot (Fig5.1) generated when one of the features in X axis is plotted against the target variable on Y axis. This is only sample of how a decision tree regressor splits up or finds the intervals in the data which can be used to predict the target value given the feature in a particular interval is given. Taking the example of the image below we find that the whole data has been divided into 8 intervals. So, for example the interval between 10 and 15 on the x axis would give fairly the same result on the y axis. Similarly, the interval between 25 and 35 is very big and also has fairly the same value on the y axis. So, exploring and using a decision tree's power to draw a relation and an interval in the pattern of data can really help in predicting the outcome of the analysis or the value of the target. Therefore, using DecisionTreeRegressor is one of my approaches to studying the iconicity.

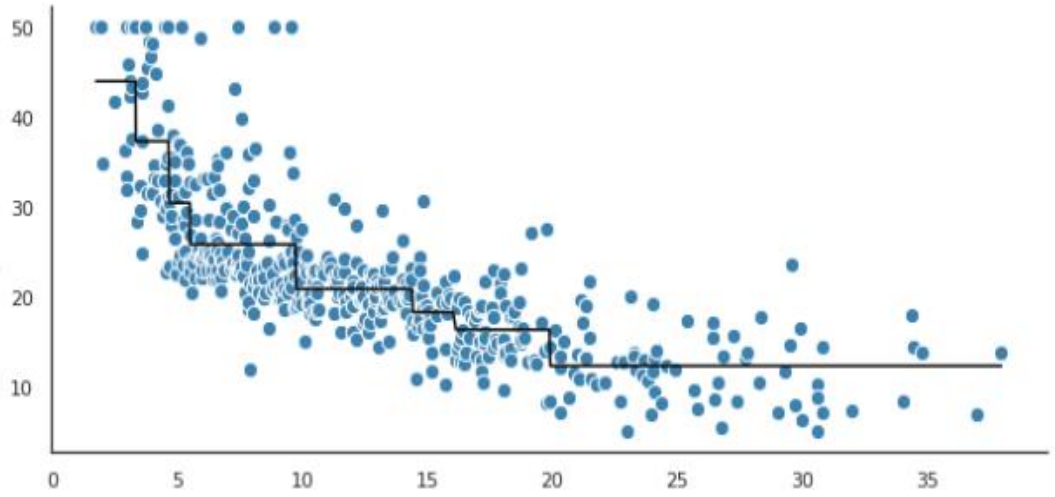
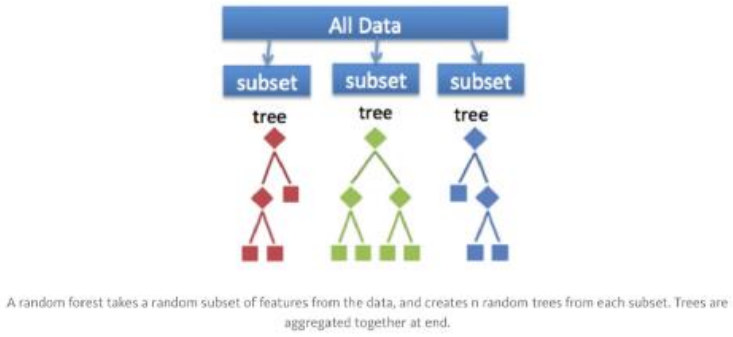


Fig 5.1: Plot of target vs feature for decision tree regressor

5.3.2 RandomForestRegressor

RandomForestRegressor is another machine learning approach that intrigued me in this research project. As discussed above in the first approach of DecisionTreeRegressor where a tree would be made based on some yes no questions the approach in this section is similar to the previous one. Ensemble is a term used in machine learning and it means combining several base or low predictive power models and getting a single model which has a higher predictive power than any individual model. In DecisionTreeRegressor a single tree is made to make a model. The problem with this approach is it is not sure on each node as to which feature should be used to split it. Some feature may give a good split while others may not. Further, the model becomes heavily dependent on the training data. The model overfits the data. This means any change in the training dataset would lead to the construction of a different decision tree and slightly different accuracy. The full potential of all the features remains unknown when only one decision tree is used.

RandomForestRegressor uses the power of ensemble learning to construct several decision trees based on different subset of features and training data. This means now several decision trees can cover almost all the features to split the data and can run themselves on different subsets of training data. This leads to reduction in variance. In case of regression problems the average of all the independent outputs from all the decision trees is calculated. For classification problems the majority of all the outputs of the decision trees are calculated. RnadomForestRegressor works on the concept of bagging. What is bagging? Bagging is also known as Bootstrap Aggregation. In Random Forest they the full independence making subsets of data. It is not only various training sets and test tests but also features on which the model should run. Below is the figure which shows how the power of RandomForestRegressor can be used to get a single model which better than any other trees.



Source:
https://miro.medium.com/max/500/0*XJx4UK_LVNTx0aaI.
Fig 5.2: RandomForestRegressor

The above figure is a good illustration of Random Forest. As it can be seen that the topmost bar represents the dataset it has been split up into various subsets. Those subsets have several trees under them. Each tree has a different shape and predictive power. In the end all trees are aggregated together for a single output.

The package that I used for RandomForestRegressor was sklearn. It can be used to import the required dependencies and packages to the code and get an object of RandomForestRegressor which can be fed with the train dataset to model it. We can do some hyperparameter tuning to adjust some parameters of the object to better adjust the model with the training dataset. The number of trees and the depth of the trees can be adjusted. When we adjust the depth of the trees then it is called pruning. It means we restrict the depth of the tree when it is seen that after a certain level of depth the tree does not remove the impurities much so that our trees do not overfit or learn the training data more and lose any general pattern of the training data itself.

5.3.3 Support Vector Machine

Support Vector Machine or SVM is another excellent machine learning algorithm which helps in classification problems. It has the ability to classify points in an n- dimensional space. The n-dimension here corresponds to n- features. Then, a line or if it is more than 2 dimensions then a hyperplane divides these points into two categories. The line or the hyperplane separating these categories of points is the decision boundary. The points on either side of this hyperplane which are closest to it are called support vectors. The distance between the support vectors and the hyperplane is called margin. Larger the margin better the classification. Support vector machine algorithm comes in supervised machine learning technique in which the features and the target are already given and the algorithm has to figure out the pattern that leads to a particular type of output. The figure below illustrates the principle of Support Vector Machine.

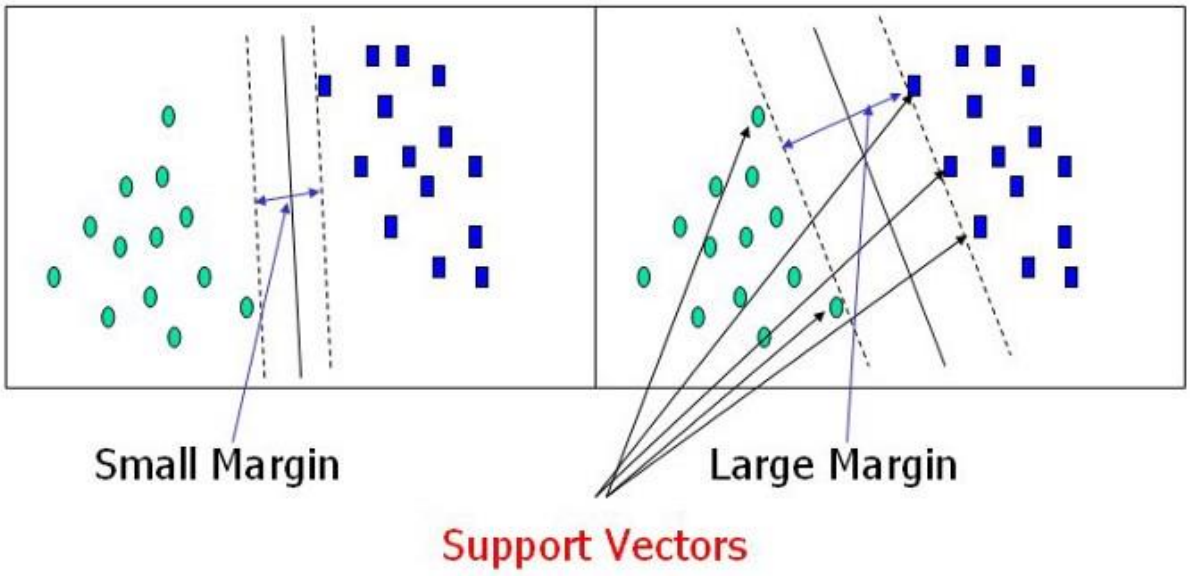
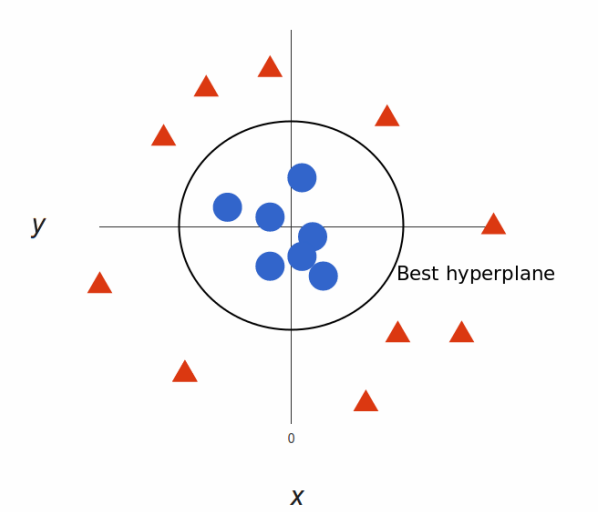


Fig 5.3: How SVM works and hyperplane and support vectors
Source:
https://miro.medium.com/max/700/0*ecA4Ls8kBYSM5nza
.jpg

The figure above shows two cases where in the first one on the left hand side of the image we have two groups of dots. The first group is round and green and the second group is blue and square. There is a hyperplane which separates these two groups and we can see two closest points which are the support vectors to this hyperplane on either side. The margin is small and hence it is not a good Support Vector Machine model. The second case is on the right hand side of the image where it can be seen that the green and round dots are well separated from blue square dots by a hyperplane. The margin between the hyperplane the support vector is good and hence the separation is maximum. Therefore, it is a better model than the previous one.

When the distribution of points is linear then the hyperplane separating the two groups of points is available but when the distribution not linear then a hyperplane separating the two groups cannot be drawn. As a result some other concept called kernel is used to separate the

distribution of points so that a unique best hyperplane can be drawn to separate the two groups. Consider the example below from the diagram.

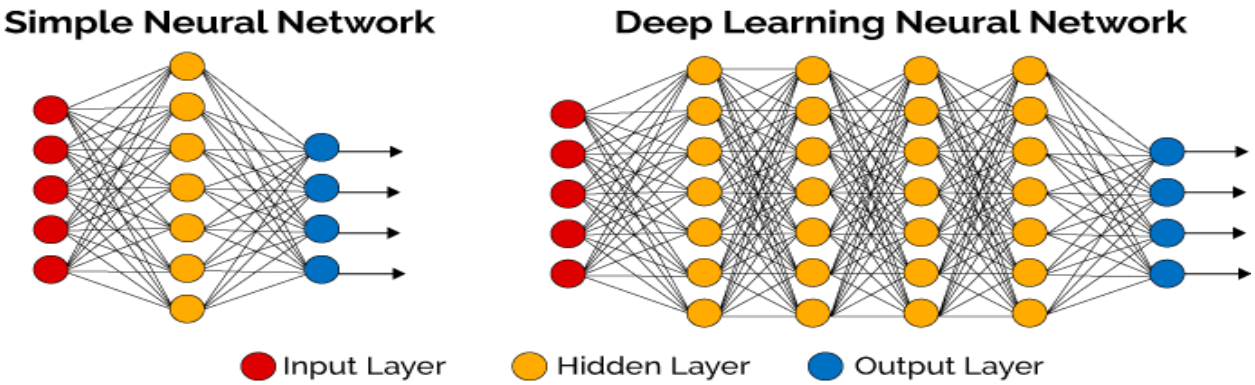


Source:https://monkeylearn.com/blog/wpcontent/uploads/2017/06/plot_circle_04.png
Fig 5.4: Illustration of non-linear distribution of points.

The figure above is an illustration of a non linear distribution of points. They cannot be separated by a line. So a circle is drawn to differentiate between the two groups that is the blue and red. Kernel concept helps here whereby points in lower dimensions are mapped onto points in higher dimensions. The above figure could also be seen as blue circles below the red triangles in the direction of z axis or in the axis perpendicular to both x and y. This is the whole concept of Support Vector Machine which can be used to make a classification model.

5.3.4 Deep Learning

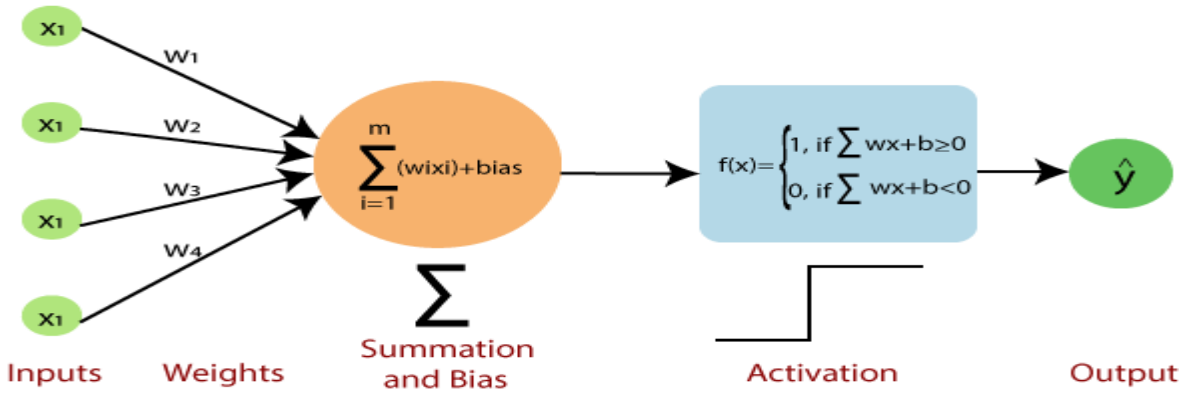
The power of artificial neural networks is being used to model many problems. The artificial neural networks can work both for classification and regression problems. Artificial neural networks attempt to imitate the working of human brain and give an output. Deep comes from the fact that when the layers increase for processing data then the smaller artificial neural network becomes big and it is called deep learning. Below is an illustration of a simple neural network and deep learning.



Source:https://cdn-images 1.medium.com/max/800/1*dnvGC-PORSoCo7VXT3PV_A.png

Fig 5.5: Illustration of a simple neural network and a deep learning network

The above figure shows the difference between a simple neural network and a deep learning neural network. As can be seen on the left side of the image there is an input layer with red circles, the middle portion is yellow, which is the processing layer and the output layer is in blue. This is called a simple neural network. A deep learning neural network contains multiple layers for processing the data before giving the output.



Source:https://static.javatpoint.com/tutorial/tensorflow/images/single-layer-perceptron-in tensorflow2.png

Fig 5.6: A perceptron showing the functioning.

A perceptron is the basic unit of any neural network. As in the above figure it takes inputs marked as x1. These inputs are the feature values of some data. Generally the inputs are assigned some weights initially to highlight each feature's importance. The inputs are then processed based on some rule. Generally the processing is done by adding the multiplication of weights and inputs and then adding a bias to the final sum (S for example). The final result is then passed through an activation function. In the example above the activation function is a step function. There are various activation functions. The step function gives an output 1 when S is zero or greater than zero. The step function gives an output 0 when S is less than zero. The output has been shown in the above figure as y. This y is then fed to the perceptrons of another layer and after several such layers a final output is received. In many mechanisms a back propagation is applied and final output is matched against the actual result. In case of any discrepancies the weights are adjusted for the inputs so that the final output matches with the actual result. This way the model trains itself and gives fairly accurate results when a test data is given.

5.4 Method 1

The first idea before applying any machine learning algorithm is to predict the classification as BEFORE and AFTER being biased. It means it would be assumed that if a machine learning algorithm predicted only BEFORE or only AFTER then what would happen. The natural question is whether the percentage of accuracy is significant or not. Considering the first case where our algorithm predicts only BEFORE then we find that its accuracy is only 46 %. This means in this dataset it is not a good guess by anyone to say that a particular sentence has the Allen relation classified as BEFORE. Next, if we consider the second case where our algorithm predicts only AFTER then we find that its accuracy is a little more than the previous case and it is 54%. This is also not significant as any random guess on what a particular sentence has the Allen relation is correct by 54 %.

The above analysis also proves that the distribution of BEFORE and AFTER classification in the whole dataset is almost equal and one does not dominate over the other otherwise
a model would learn more about one case than the other and naturally its prediction capacity would improve leading to false assumption that the model is highly accurate.

However, different algorithms have different predictive power, it is important to see how each of them behave when the same dataset is applied to them and especially the same train – test split is applied to them. The results of each of those models have been described in the following sections. It is worth noting that the accuracy improves to a great extent and hence the predictive power of the models.

5.5 Method 2

The first method which has been used is DecisionTreeRegressor. This is an obvious choice for these kinds of problems as the result column that is the ‘Iconicity’ column is having either 0 or

1. This means that it is a classification problem. DecisionTreeRegressor is a model that builds a decision tree based on some yes-no questions. The output is not exactly one of the categories but it gives a probability value or closeness to one of the values that can be in the result. Later the threshold can be decided as to after which value it can be considered as 1 or otherwise 0. In the python code the iconicity column is made of the fact that if any event or time comes before another event or time on the textual side then it is treated as 0 or classified as BEFORE. If on the other hand if any event or time comes after another event or time on the textual side then it is treated as 1 or classified as AFTER. The python code for this contains a separate method which serves the purpose of carving this information out of the xml files and also accordingly marking the conditions as 0 or 1. For DecisionTreeRegressor sklearn package has been used to get a DecisionTreeRegressor object in python and model it according to the training data. The split of the dataset into train and test dataset was done. The train dataset on which training can be done on a model was 70 % of the whole dataset. The test dataset to check or validate the performance of the model was 30 % of the whole dataset. Again for the train test split sklearn's model_selection package was used. First, the training data is fed into the model's object and then the model trains itself according to data and learns the patterns in the data. For it we hold an object of the model. After it trains it is tested with the test data. The model predicts the output in a number which lies between 0 and 1 both 0 and 1 included. Then, the actual result is matched with the results predicted and then the normal accuracy is found out in terms of the percentage of correct predictions or matches with the actual result. The figure below illustrates the output of the prediction.

```
array([1.        , 1.        , 1.        , ..., 0.55555556, 1.
       0.53846154])
```

Fig 5.7: Illustration of output from DecisionTreeRegressor

As can be seen from the figure above some values have been predicted to be 1 and other values are having decimal places in them and lie between 0 and 1.

The model was run on different train-test split. The first option is to vary the proportion of train-test split. For example, trying to get an 80 % share of train data and 20 % share of test data is one such option. However, different such shares of train and test split did not prove to be as efficient as the 70 % - 30 % split. Another option was to get different train test dataset from the same proportion of split, It means keeping the shares of train and test dataset as 70 % - 30 % run the model several times to get the optimum split. It works as when train and test split is done it is completely random. Every time the same code is run for train and test split then every time we get a new set of data in the train part and a different set of data in the test part. This is interesting as such different splits can make model learn more in one case and a little less in other cases. To get a hold of one such fixed set of train and test dataset a parameter in the train – test split is used and adjusted. The parameter is called random_state. The optimum value of this parameter can be found so and that value can be used to get a train-test split which would be good for the model top learn the pattern in data more. For one such run it was found to be 452. For that value the results have been achieved. Obviously, there can be more optimum values and can be found out.

After having done all that the result is calculated. It is 75.22 % in the case of DecisionTreeRegressor. This is far better than the figures achieved for BEFORE and AFTER when a biased classifier is used as described in Method 1. The increase is about 29.22 % if predicted only BEFORE and the increase is about 21.22 % when predicted only AFTER. This is significant as now the model has learned the annotation patterns done by the human annotator and is able to say whether the semantic of an event or the order in which the events happened for real is mirrored in the text or not. Thus, we can understand the iconicity of an event by doing some analysis and running some machine learning algorithms on the data.

5.6 Method 3

RandomForestRegressor is the next choice to look into. As described in the background work the power of random forest is immense in getting a better final predictor than the individual predictors therefore it is evident that this method should be used for this research project.

The sklearn package provides the object of RandomForestRegressor. This object can be modeled with the train data. The same mechanism of train-test split is used for RandomForestRegressor. In fact the same split dataset is used for RandomForestRegressor as is used for DecisionTreeRegressor. This is necessary to see the behavior of all the methods for the same set of train and test split. The random_state used was again 452. At the first attempt a normal object was used for modeling the data. It means there was no specific choice of depth of the trees or the number of trees. The result gave a slight improvement over DecisionTreeRegressor. The accuracy was 76.20 %. As compared to the accuracy of the biased classifier it is a good improvement as the accuracy is now 30.2 % more than if it had only predicted BEFORE. It is 22.2 % more than if it had predicted only AFTER. It is also 0.98 % more than DecisionTreeRegressor which is definitely a little improvement than DecisionTreeRegressor. The output received from the model of RandomForestRegressor lied in the range of 0 to 1 with both 0 and 1 included.

For both DecisionTreeRegressor and RandomForestRegressor the predictions were in the range of 0 to 1 with 0 and 1 included. Therefore, a threshold value had to be determined above which it had to be considered 1 and below which it had to be considered 0 as it is a classification problem and it has to be decided whether a particular combination of annotations is classified as BEFORE or AFTER. For DecisionTreeRegressor first the threshold was adjusted and values from 0.1 to 0.9 were tried for marking those values as 1 which are above these threshold values. The result received was worse than 0.5. This proved that 0.5 is the best threshold value. Similarly, for RandomForestRegressor the threshold value was adjusted to a range of values from 0.1 to 0.9 to see which can give a better result. In this case also 0.5 was the optimum value and for this value the prediction accuracy was the maximum.

5.7 Method 4

SVM or Support Vector Machine was the next method used to model the data. As Support Vector Machine has a unique ability to draw a line or a hyperplane between the two groups of data it is useful in classification problems. Using the same technique as above the train-test split was the same as used in DecisionTreeRegressor and RandomForestRegressor an object from the sklearn package is taken in the python code. The object is used to get trained by the train dataset. The trained model was tested against the test dataset and the output was observed. This also showed a remarkable trained model and the accuracy was much better than the biased classifier. The accuracy was 75.67 %. It is 29.67 % better than the BEFORE biased classifier and 21.67 % better than the AFTER classification. Again it proves that the there is a pattern of annotations in the document which can be studied the iconicity can be determined. In SVM the output is 0 and 1 and not in a range like DecisionTreeRegressor and RandomForestRegressor. It means a Support Vector Machine model classifies the data as BEFORE or AFTER in a clear manner. Therefore, in this no threshold value is required. With this some hyper parameters can be adjusted to make the model better. This is the simplest approach that can be taken to use Support Vector Machine.

All the above methods relate to some machine learning algorithms but the power of neural networks can be explored to understand the behavior of the annotated document when its annotation are read.
Deep learning is another option which can be used to understand the iconicity from a different angle.

5.8 Method 5

Deep Learning was used to train the dataset. The package used in the python code is keras. The sequential keras model is chosen and the Dense object in the python code is used to build the layers of the Deep Neural Network. The same train and test split was used to train the

model as was used to train the models above. The number of epochs was 100. One epoch is when the entire dataset is passed forward and backward through the neural network. The batch size used was 32 for each of the layers. Batch size is the number of training elements that pass through the whole neural network forward and backward in one iteration. Well in this research project the batch size can be the whole training dataset as the whole dataset itself is very small still to increase the number of iterations and epochs the training dataset can be divided into batches for proper adjustment of weights in the neural network. After 100 epochs it could be seen that the model was ready and could be tested against the test dataset. The accuracy achieved was again remarkable and it was 76.05 %. This is still better than 46 % of BEFORE biased classifier and 54 % of AFTER biased classifier. The image below shows the run of a deep neural network with the epochs.

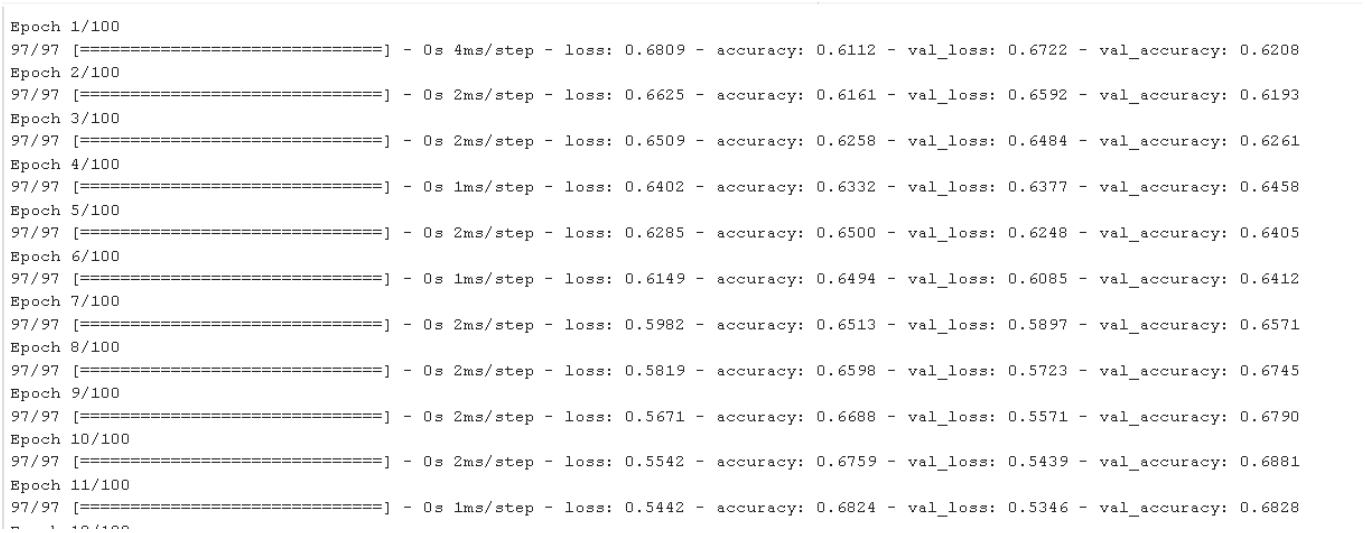


Fig 5.8: Illustration of run of deep neural network

5.9 Analysis of Individual Allen Relations

After having run the machine learning algorithms and deep learning on the whole dataset it is important to see how these machine learning algorithms and deep learning behave when they are modeled on individual Allen relations. The approach above was on the whole dataset but if only individual TimeML tags are considered then the question arises whether these algorithms learn the same pattern of data or they become better or degrade in performance. For the analysis of individual Allen Relations the code was changed a little to make the dataset first. Individual files containing a particular tag could be generated through the code. For example, one file would contain only BEFORE tags while another file would contain only AFTER tags. By this a fine grained result can be derived. The same method of loading the files and splitting the dataset into train and test split was done. The split ratio was kept the same that is 70 % train data and 30 % test data. All the models were run and the best performing model for each of the Allen relation was chosen. The result can be summarized below in a table.

TimeML tag	Algorithm	Accuracy
BEFORE	RandomForestRegressor	86.6 %
AFTER	Support Vector Machine	74 %
BEGINS	RandomForestRegressor	77.7 %
ENDED_BY	RandomForestRegressor	84.6 %
IBEFORE	Support Vector Machine	72.7 %
IS_INCLUDED	Support Vector Machine	72.4 %
SIMULTANEOUS	Support Vector Machine	64.2 %

Table5.1: Table showing individual analysis of TimeML tags

From the above table it can be seen BEFORE tag was learned more by the model and therefore it predicted an accuracy of 86.6 %. There are two major reasons as to why the accuracy is good for BEFORE as compared to other tags. First, the pattern of annotation was good so that the model learned it properly. The second is the number of BEFORE tags was more than the other tags so that the model got a good number of training data to train on. The accuracies of other TimeML tags lie in the range of 70 % to 84 %. The accuracy of SIMULTANEOUS tag is 64.2 %. This suggests that either the SIMULTANEOUS tag was not annotated properly or the number of training data was very less.

5.10 Adjustment for TLINKs and Removal of Biased Annotation

The analysis shown earlier give a good description of the annotation and the predictive power of all the models based on the annotation. However, the fine grained analysis of individual tags shows a large proportion of BEFORE tags. That is 86.6 %. When carefully observed the dataset contains a lot of biased annotation. It means the human annotator who was responsible for annotating the documents might have been a little biased towards a particular kind of tag. In this case it is the BEFORE tag which has more occurrences in the document than it should have had. There are other tags as well which show some biased annotation but the BEFORE tag has a greater share in it. BEFORE and AFETR are inverse of each other. Therefore, if someone is biased towards a particular kind of tag then he has to just swap the arguments and change from one type of tag to its inverse. It is explained as follows through an example. Let us consider an annotated part of a document in the following example. The sentence “More problems in Hong Kong for a place, for an economy, that many experts thought was once invincible.” has been annotated.

More
<EVENT eid="e153" class="OCCURRENCE">problems</EVENT>
in Hong Kong for a place, for an economy, that many experts

<EVENT eid="e11" class="I_STATE">thought</EVENT>
was once
<EVENT eid="e376" class="STATE">invincible</EVENT>.

<MAKEINSTANCE eventID="e376" eiid="ei438" tense="PAST" aspect="NONE" polarity="POS" pos="ADJECTIVE"/>
<MAKEINSTANCE eventID="e153" eiid="ei383" tense="NONE" aspect="NONE" polarity="POS" pos="NOUN"/>
<TLINK lid="l53" relType="BEFORE" eventInstanceID="ei438" relatedToEventInstance="ei383"/>

As can be seen that in the TLINK tag the relType is BEFORE. The eventInstanceID is having a higher MAKEINSTANCE id ei438. The relatedToEventInstance is having a lower MAKEINSTANCE id ei383. The semantic of the sentence says that eventID e376 comes before e153. It can also be said that e153 comes after e376. The relType can be changed to AFTER only when eventInstanceID has the value ei383 and relatedToEventInstance has the value ei438. This leads to thinking that just swapping the values between eventInstanceID and relatedToEventInstance can change the relType and make the document look more dominated with one type of tag. The convention followed should be the same when deciding the relType. The lower order MAKEINSTANCE id should be in eventInstanceID and the higher order MAKEINSTANCE id should be in relatedToEventInstance.

In this example adjustment is needed and the TLINK should look like
<TLINK lid="l53" relType=”AFTER" eventInstanceID=" ei383 " relatedToEventInstance=”ei438"/>

Let us consider another example where adjustment is not needed. The sentence ”The financial assistance from the World Bank and the International Monetary Fund are not helping.” has been annotated in the following way.

The financial

<EVENT eid="e3" class="OCCURRENCE">assistance</EVENT>

from the World Bank and the International Monetary Fund are not

<EVENT eid="e4" class="OCCURRENCE">helping</EVENT>.

<MAKEINSTANCE eventID="e3" eiid="ei377" tense="NONE" aspect="NONE" polarity="POS" pos="NOUN"/>

<MAKEINSTANCE eventID="e4" eiid="ei378" tense="PRESENT" aspect="PROGRESSIVE" polarity="NEG" pos="VERB"/>

<TLINK lid="l1" relType="BEFORE" eventInstanceID="ei377" relatedToEventInstance="ei378"/>

In the above example no adjustment is required as BEFORE on the textual side is represented in the sequential order in the TLINK tag.

The shares of different TimeML tags in the whole corpus of data is visualized in pie charts in the following. The pie charts show how a particukar tag looked like textually. It basically gives a share of a tag when classified as BEFORE and AFTER. It an event comes prior to an event on the textual side then it is considered as BEFORE and if it come after an event then it is considered as AFTER. The visulaization shows how each of the tags look like when classified as BEFORE and AFTER.

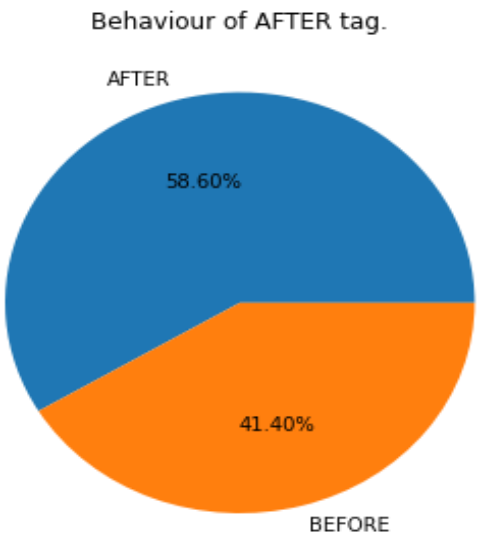


Fig 5.9: Share of AFTER tag when classified textually as BEFORE and AFTER

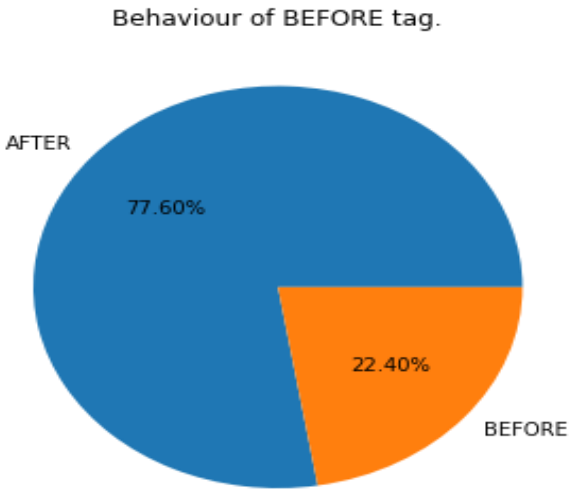


Fig 5.10: Share of BEFORE tag when classified textually as BEFORE and AFTER

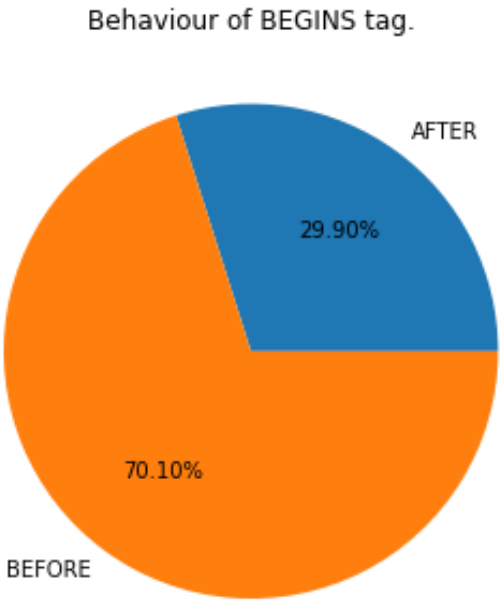


Fig 5.11: Share of BEGINS tag when classified textually as BEFORE and AFTER

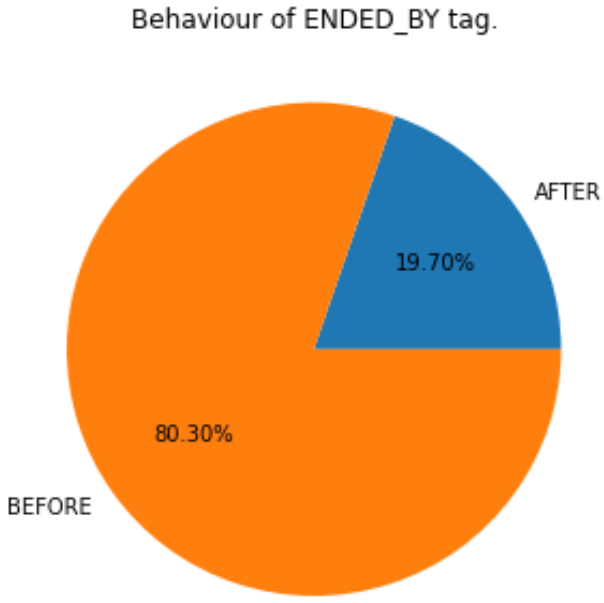


Fig 5.12: Share of ENDED_BY tag when classified textually as BEFORE and AFTER

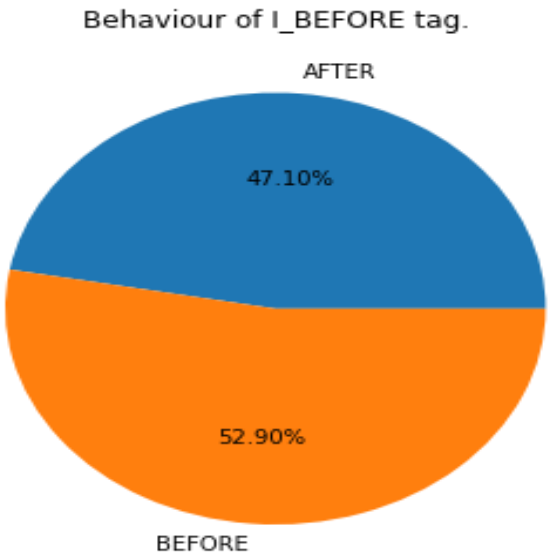


Fig 5.13: Share of I_BEFORE tag when classified textually as BEFORE and AFTER

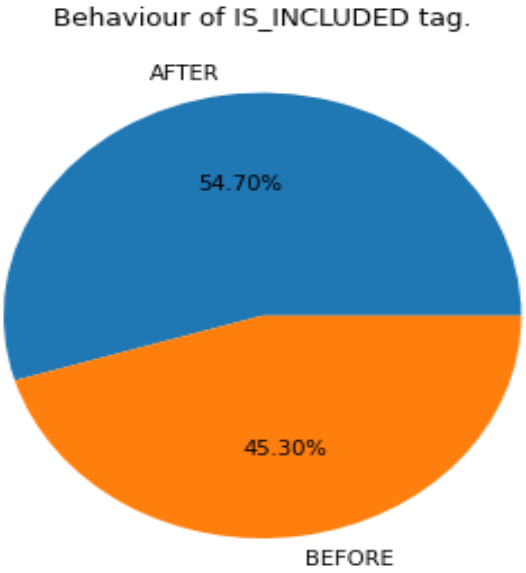


Fig 5.14: Share of IS_INCLUDED tag when classified textually as BEFORE and AFTER

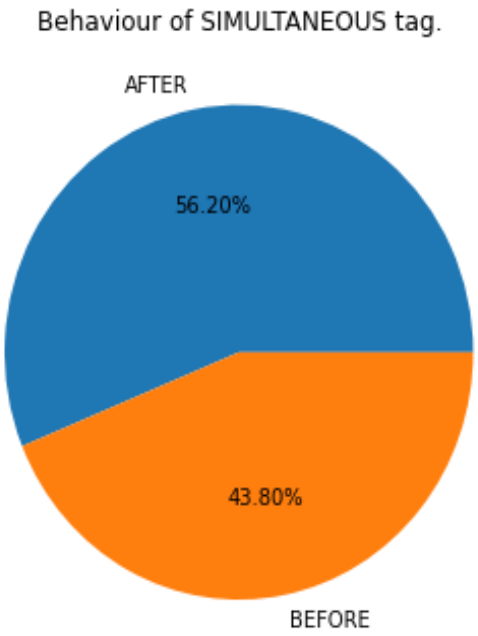


Fig 5.15: Share of SIMULTANEOUS tag when classified textually as BEFORE and AFTER

As can be seen from the image above that the BEFORE tag is behaving more like an AFTER tag which again proves that the document was annotated being biased towards the BEFORE tag. To fix this and see how many such adjustments are needed a little change in the code was done to introduce another column in the csv file by the name AdjustedRelType. In this column if any adjustement is required for any TLINK tag then the inverse tag is written else a ”No” is written. The following figures show how the shares of individual tags changed after the adjustment is done. In the image with the adjusted link new TimeML tags can be found which should have existed in the documents it they were not biased. The new tags are the inverse of the tags present in the documents. SIMULTANEOUS tag has no inverse and hence it is the same before adjustment and after adjustment.

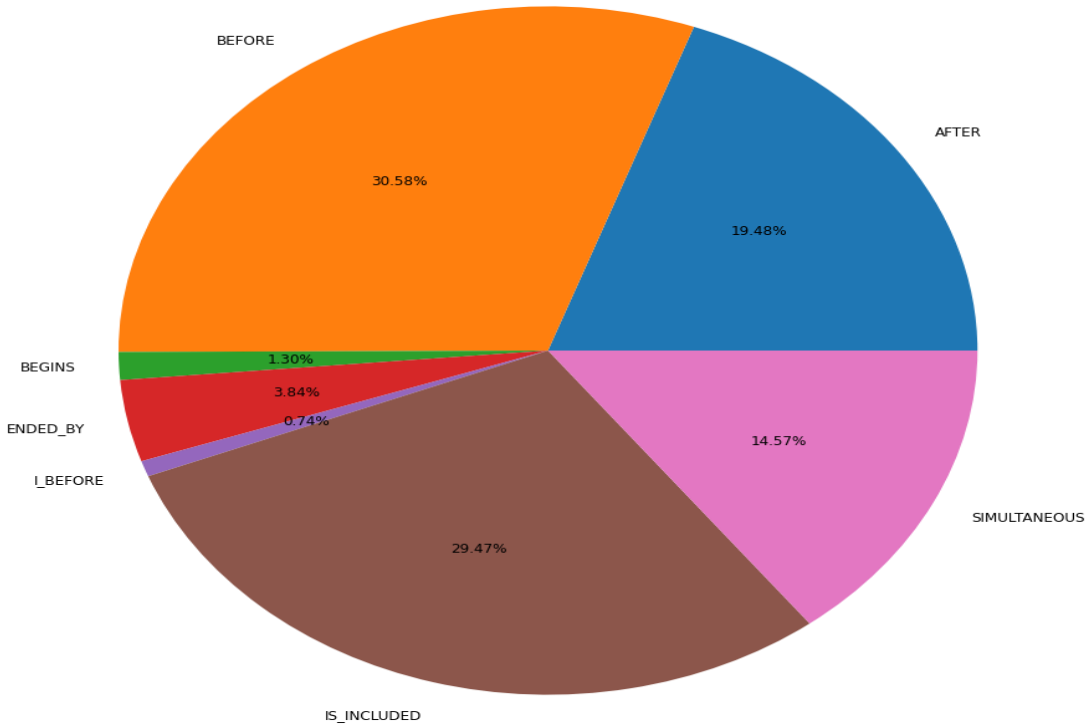


Fig 5.16: Share of TimeML tags in whole corpus dataset

The above figure shows the shares of Allen relations TimeML tags in the whole corpus dataset. As can be seen the BEFORE tag has 30.58 % and the AFTER tag has 19.48 %. It is evident from the analysis above that the BEFORE tag is a biased tag and the whole dataset needs to be adjusted for removing the biased annotations. After removing the biased annotation and making the adjustment the new shares of the TimeML tags is shown in the figure below which shows the comparison between the shares of TimeML tags before adjustment and after adjustment. The first pie chart shows the shares after adjustment and the pie chart following it shows the shares before adjustment. It can be observed that the share of BEFORE tag reduced to 17.77 % from 30.58 %. The share of AFTER tag increased from 19.48 % to 32.30 %. Another significant split can be observed in IS_INCLUDED tag as well. It got split into IS_INCLUDED and INCLUDES with 13.40 % and 16.07 % shares respectively. It may be noted that the area covered by a tag and its inverse is the same as covered by one of the tags before adjustment.

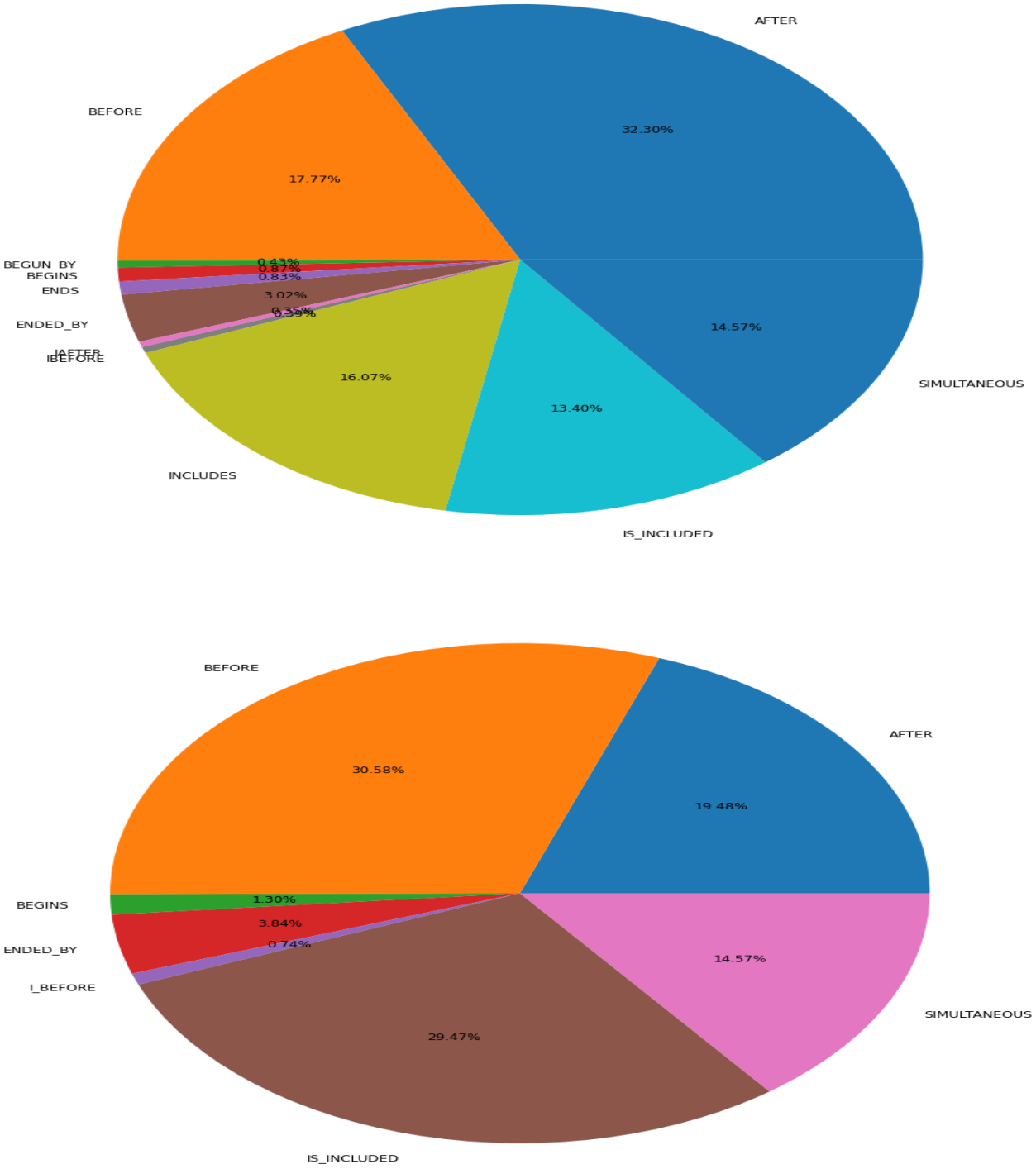


Fig 5.17: Comparision of TimeML tags shares before adjustment and after adjustment

6. Future work

The research above is a novel approach in terms of studying the iconicity of a sentence. The methods used are related to machine learning algorithms and using the power of these algorithms to predict the category (BEFORE and AFTER) in which a sentence falls. Further a fine grained result of all the individual tags is done to understand the behavior of the algorithms to these tags. The code can be improved to better the predictive power of the models. This is important as there are many hyperparameters that can be tuned to make the models better. In this research project four models have been used to study the measure of iconicity. However, they all were close to each other in terms of accuracy. It has to be ascertained if all the models are required else the measure of iconicity can be studied by just one model. The second work that can be done here is the cleaning of data properly. As most of the time was spent in understanding how to study the iconicity of data in a given document the cleaning of dataset in the csv file before applying it to the machine learning models could be done better. This is a future work which can make better models which can predict the iconicity well. The dataset on which the models run has some features. It can be estimated which features are the most important but inclusion of more features in the dataset can improve the predictive power of the models. This can happen as when a feature is found which is more correlated with the result then it can explain the behavior of the output column better. There may be other important features which can explain why a certain sentence was spoken in a particular way. The results derived can be used to make a better annotator. The accuracy achieved from the models is significant and therefore it can be used to understand the power of the annotation that is being used to study the iconicity of the expressions. If a new annotator scheme is used and if it gives better result then it can be used

instead of the previous one. In [4] a study of iconicity and positioning of temporal and main clauses has been done using logistic regression. The study of this research project can be compared with the approach and seen which one is better or how both can be explored to understand the measure of iconicity. The choice of using better annotator can be used in information extraction. Information extraction and Natural Language Processing communities are interested in temporal expressions. To give a quantitative analysis of the pattern of annotations can really help in progressing through defining better ways to understand the temporal relation between two temporal events. It is important here to understand that the dataset for studying and applying the methods described in this paper needs to be enhanced. Larger the dataset better the learning of the models. It should also be taken care of that the number of each Allen relations should be fairly the same else if one Allen relation outnumbers the others by a large amount then it may give a biased learning to the models. So a proper choice of dataset should be done in the future.

7. Conclusion

The purpose of this research project was to study the measure of iconicity. The project is successful in doing so and giving a novel approach to understand the iconicity. It takes into account the annotations proposed and done in a document to study the temporal relations between two events, two times or an event and a time. The dataset used is the TimeBank corpus. As the annotations were already done and all the relations were established between two temporal events the challenge was to study the efficiency of the annotation scheme. The efficiency would show how efficiently the annotation scheme is able relate two temporal events. It was also seen that the documents were annotated by using TimeML tags which represent the Allen Relations between two temporal events. As there are thirteen Allen relations that can exist between two temporal events they had to be reduced down to only two Allen relations that are BEFORE and AFTER. This would simplify the understanding of temporal relation as only two Allen relations had to be studied. Both these challenges were

worked upon and I could carve the data out of the annotated documents through a python code and apply machine learning algorithms to it. I analysed that a biased classifier had the accuracy of around 50 % in determining whether a particular sentence is classified as BEFORE or AFTER on the textual side. The machine learning algorithms gave good results and the accuracy of prediction rose over 75 %. This shows that the annotation scheme can be analyzed through machine learning algorithms and the iconicity of a sentence can be determined given a set of features. Having done this analysis using different machine learning algorithms and deep learning it is evident that there is a pattern in the annotation scheme and the cases for which the prediction fails to give the correct result can be checked and rectified. The failed cases suggest improvement on the code side as well as on the side of annotation scheme. It may be possible that for a particular TimeML tag the annotation scheme is poor to properly annotate the events. Therefore, in such cases improvements may be needed to properly annotate the temporal relations. In the fine grained analysis of individual TimeML tags the models learned the variation in data and could predict the classification category to which any sentence fell. The models gave the best prediction in the case of BEFORE tag and worst prediction in the case of SIMULTANEOUS tag. It is sure that either BEFORE tag is in large number and SIMULTANEOUS tag is in low number. It is also true that the annotation scheme is good in case of BEFORE but for SIMULTANEOUS the annotation scheme is not good and is unable to predict the the classification of sentences as BEFORE and AFTER. This leads to a conclusion that a good number of data for all the TimeML tags is needed to ascertain which tags are properly explained by the models. This also explains how efficient the pattern of relation between temporal events has been studied. It means if the pattern of temporal relations is studied well then a proper annotation scheme can be made. This research project does not come only with analyzing the dataset but also making a dataset out of the annotated document. The python code written for making a dataset suitable for machine learning can be taken as a platform for making a similar dataset for any annotation scheme. Another important conclusion that can be derived from this research project is that the biased annotation and “relType” in TLINK tags. This research project highlights that the

relType in the TLINK tags should be decided keeping the temporal events in the sequential order as they occur in the text. This helps in removing the biased annotation of the human annotator who might be biased towards a particular type of TimeML tag and the document might look like filled more with one type of TimeML tag and very less its inverse. It also concludes that the presence of all the tags can help in learning the iconicity better. The models can learn the variation in data better. Though this fact was known at a much later stage in the project it is a welcome sign for anyone who works on this project or the human annotator who annotates the documents. This project also opens up possibilities to understand the iconicity in other languages. The variation can be studied. Finally, to sum up the research project is a success and has used a novel approach in studying the iconicity. It has been able to make a dataset and apply machine learning algorithms to it. The results derived were good and this research can be extended to achieve better results.

Bibliography

[1] Styler WF 4th, Bethard S, Finan S, et al. Temporal Annotation in the Clinical Domain. Trans Assoc Comput Linguist. 2014;2:143-154.

[2] Derczynski, Leon & Gaizauskas, Rob. (2013). Empirical Validation of Reichenbach's Tense Framework. 71-82.

[3] Pustejovsky, J., Castaño, J.M., Ingria, R., Saurí, R., Gaizauskas, R., Setzer, A., Katz, G., & Radev, D.R. (2003). TimeML: Robust Specification of Event and Temporal Expressions in Text. New Directions in Question Answering.

[4] Diessel, H. (2008). Iconicity of sequence: A corpus-based analysis of the positioning of temporal adverbial clauses in English.

[5] Perniss, P., Thompson, R. L., & Vigliocco, G. (2010). Iconicity as a general property of language: evidence from spoken and signed languages. Frontiers in psychology, 1, 227. https://doi.org/10.3389/fpsyg.2010.00227

[6] Zhang, Yuchen & Xue, Nianwen. (2018). Structured Interpretation of Temporal Relations.

[7] Pustejovsky, James & Hanks, Patrick & Saurí, Roser & See, Andrew & Gaizauskas, Rob & Setzer, Andrea & Radev, Dragomir & Sundheim, Beth & Day, David & Ferro, Lisa & Lazo, Marcia. (2003). The TimeBank corpus. Proceedings of Corpus Linguistics.

[8] Ferro L., Gerber L., Mani I., Sundheim B., Wilson G. (2003) TIDES 2003 Standard for the Annotation of Temporal Expressions.

[9] Allen, J.F. (1984). Towards a General Theory of Action and Time. Artif. Intell..

[10] Altenberg, Bengt1984Causal linking in spoken and written English.Studia Linguistica38, 20–69.Backhaus, Klaus, Bernd Erichson, Wul¤ Plinke, and Rolf Weiber2006Multivariate Analysemethoden. Eine anwendungsorientierte Einfu¨hrung. Ber-lin: Springer. [11th. edition].

[11] Bethard, S., Derczynski, L., Savova, G., Pustejovsky, J.,and Verhagen, M. (2015). Semeval-2015 task 6: Clinical tempeval. InSemEval@ NAACL-HLT, pages 806–814.

[12] Yoshikawa, Katsumasa & Riedel, Sebastian & Asahara, Masayuki & Matsumoto, Yuji. (2009). Jointly Identifying Temporal Relations with Markov Logic.. 405-413. 10.3115/1687878.1687936.