CHENNAI MATHEMATICAL INSTITUTE

Natural Languages and Not-So-Natural Processing

by

Kumar Madhukar

A thesis submitted in partial fulfillment for the degree of Master of Science (Computer Science)

in the

Department of Computer Science

June 2008

Declaration

This thesis is based on the work done by the author at Valuepitch Interactive IT Services Pvt. Ltd., Mumbai, under the guidance of Prof. Pushpak Bhattacharyya (CSE Dept., Indian Institute of Technology Bombay). All references to the published work of others have been clearly attributed.

Author

Kumar Madhukar

Certified by

Prof. Pushpak Bhattacharyya
Thesis Supervisor
IIT Powai

Examined by

Raj D'Souza
CTO, Valuepitch

Acknowledgements

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I'd like to thank my advisor, Professor Pushpak Bhattacharyya for guiding me through this. Without his brilliant theoretical ideas and computational expertise, this thesis would have been a distant dream. It has been a pleasure to work with and learn from such an extraordinary individual.

I would also like to thank Mr. Venkat Ramna (CEO, Valuepitch) and Mr. Raj D'souza (CTO, Valuepitch) for giving me an invaluable opportunity to work on challenging and extremely interesting problems over the past few months. They have always made themselves available for help and advice, and have enriched my graduate life in many ways.

I would also like to thank my co-advisor, Prof. Madhavan Mukund for his constant guidance throughout my undergraduate and graduate years. Words cannot express the gratitude I owe him.

My colleagues at Valuepitch have been very helpful in providing a conducive atmosphere throughout.

I owe my deepest thanks to my parents who have always stood by me and guided me through my career, and have pulled me through against impossible odds at times.

It is impossible to remember all, and I apologize to those I've inadvertently left out.

Lastly, thank you all and thank God!

Introduction

Natural language processing (NLP) is a subfield of artificial intelligence and computational linguistics. Computational linguistics (CL) is a discipline between linguistics and computer science which is concerned with the computational aspects of the human language faculty. It belongs to the cognitive sciences and overlaps with the field of artificial intelligence (AI), a branch of computer science that is aiming at computational models of human cognition.

Contents

Chapter 1

Natural Language Processing : An Overview

1.1 Definition

A 'natural language' (NL) is any of the languages naturally used by humans, i.e. not an artificial or man-made language such as a programming language. 'Natural language processing' (NLP) is a convenient description for all attempts to use computers to process natural language. NLP includes:

- Speech synthesis: although this may not at first sight appear very 'intelligent', the synthesis of natural-sounding speech is technically complex and almost certainly requires some 'understanding' of what is being spoken to ensure, for example, correct intonation.
- Speech recognition: basically the reduction of continuous sound waves to discrete words.
- Natural language understanding: here treated as moving from isolated words (either written or determined via speech recognition) to 'meaning'. This may involve complete model systems or 'front-ends', driving other programs by NL commands.
- Natural language generation: generating appropriate NL responses to unpredictable inputs.

- Machine translation (MT): translating one NL into another.

1.2 Origins

- The idea of using digital computers in NLP is 'old', possibly because one of the first uses of computers was in breaking military codes in the second world war. Some computer scientists seem to have thought that Russian (for example) is just English in a different code. In which case, since codes can be broken, so can Russian. This idea assumes there is a common 'meaning base' to all natural languages, regardless of their surface differences. The overwhelming consensus among linguists is that this is simply not true.
- 'Artificial Language Processing', in the form of compilers and interpreters for programming languages, was a key component in the success of digital computers from their earliest days. This success undoubtedly encouraged research into NLP (and also encouraged an optimistic approach).

There have been cycles of optimism and pessimism in the field of NLP (we are possibly in a more optimistic phase at present); although some very real advances have been made, the target of a general NLP system remains elusive. Historically, computer scientists have often been far too over-optimistic about NLP, probably for some of the reasons noted above. It is thus important to be clear from the outset exactly why the task is difficult.

It is also important to note that there are differences between natural languages. More work has probably been done on English than on any other language, largely because of the importance of American researchers, although there are very active workers in Europe and Japan. However, English is in some ways an untypical language, as it uses few inflections and relies heavily on word order. Textbooks and other introductory sources written in English rarely contain adequate discussions of NLP for languages with markedly different grammatical structures.

We can distinguish at least three distinct 'levels' in processing NL:

- Sounds

- Grammar
- Meaning

Each can be divided into two or more sublevels, which need not concern us here. What I want to do in this brief introduction is to illustrate some of the problems in processing each level.

1.3 Speech

Consider these three words, spoken by a native English speaker: *input, intake, income*. It's clear that all three words contain the element *in* with the same meaning. To input is to put something in; the intake of a water pump is the place where water is taken in; your income is the money that you earn, i.e. that comes in.

Is the element *in* pronounced the same in all three words (by a specified speaker, say someone from the south of England)? Careful listening will show that it is not. The word *input* is pronounced as if spelt *imput*, whereas *intake* is pronounced as spelt. If we let *N* stand for the sound usually spelt *ng* in English (e.g. in words like *sing* or *singer*), then income is pronounced *iNcome*.

I specified native English speakers from the south of England because many speakers of Scots English do **not** behave in this way; instead they consistently pronounce the first element of all three words as it is spelt, i.e. as in (as may all English speakers when speaking slowly and emphatically).

Interestingly, English speakers are generally quite unaware of these differences, both in their own speech and the speech of others. This is not because they cannot distinguish between the three sounds *m*, *n* and *N*. The three words *rum*, *run* and *rung* differ **only** in these three sounds and are quite distinct to all native English speakers.

Another example of the same kind of phenomenon occurs in the plurals of English nouns. Consider the words *cat*, *cats*, *dog* and *dogs*. A native English speaker explaining how plurals are formed is likely to say something like "you add an *s* sound to the end." Careful listening will show that *cats* is indeed pronounced with a *s* sound, but *dogs* is not: it ends with a *z* sound. Yet as with the previous example, English speakers don't

normally notice this difference. Again it isn't because they can't distinguish between *s* and *z* since *Sue* and *zoo* or *hiss* and *his* differ only in their *s* and *z* sounds.

The conclusion is that the sounds that native speakers 'hear' are not the sounds they make. This is important in both speech synthesis and speech recognition. When generating speech from written text, a synthesizer must not turn every *n* or *s* into an *n* or *s* sound, but must use more complex rules which mimic the native speaker. Similarly a speech recognition system must recognize the sounds in *rum* and *run* as distinct, but must **not** decide that *imput* and *input* are different words.

Even more awkward is the fact that this behaviour operates across word boundaries. If we try saying these sentences quickly and in an 'informal' style:

When playing football, watch the referee.
When talking about other people, watch who's listening.
When catching a hard ball, wear gloves.

We'll find that we say whem, when and wheN respectively. This means that

- speech synthesis system will not sound right if it simply uses pre-recorded words, since how these are pronounced changes depending on neighbouring words
- a speech recognition system must treat the three pronunciations of when as the same while distinguishing between the same sounds in *rum*, *run* and *rung*.

Even this example doesn't capture all the complexities. A native speaker of American English pronounces the *t* and *d* in the words *write*, *writer*, *ride*, *rider* differently from a native speaker of English English. If we ignore the very slightly different pronunciation of the *d*, the words will be pronounced roughly as *write*, *wrider*, *ride* and *rider*. That is, *write* and *ride* are clearly distinguishable whereas *writer* and *rider* are pronounced **exactly** the same. Yet a native speaker of this dialect doesn't 'hear' this. The actual sounds used in the sentence *I'm a writer* and *I write books* will be something like *I'm a rider* and *I write books* but will cause no confusion whatsoever; a listener who is used to this dialect will hear *writer* not *rider*.

Conclusion: to recognize words correctly requires simultaneous processing of sound, grammar and meaning.

1.4 Grammar

Grammar in this context refers to both the structure of words (morphology) and the structure of sentences (syntax). I'm only going to consider some syntax examples here.

Compilers process the syntax of a programming language without needing to understand it. Can we write similar programs to process the syntax of a NL without processing the semantics, i.e. without trying to understand what it means?

Let us consider:
'Twas brillig, and the slithy toves
Did gyre and gimbal in the wabe:
All mimsy were the borogoves,
And the mome raths outgrabe.

Without knowing what Lewis Carroll meant by some of these words, i.e. without being able to perform semantic processing, considerable syntactic processing is possible. Thus you can answer questions such as:

What were the toves doing? – They were gyring and gimballing.
Where were they doing it? – In the wabe.

(Notice that the answers may involve morphological changes to words in the original, e.g. gyre becomes gyring.)

Chomsky's famous
Colourless green ideas sleep furiously.
is another example of a syntactically correct sentence with erroneous or incomplete semantics. Again we can answer questions:

What were the ideas doing? – They were sleeping furiously.

Since people can answer such questions without needing to understand them, it seems at first sight plausible that a program could be written to do the same.

However, consider these sentences of Carroll's poem:
One, two! And through and through
The vorpal blade went snicker-snack!

Syntactically the sentence is ambiguous. It could be equivalent to

Snicker-snack went through and through the vorpal blade.

or

The vorpal blade went through and through (something) snicker-snack.

depending on whether snicker-snack is taken to be a (plural?) noun or an adverb.

More serious examples make the same point: in NLs, syntax **cannot** be processed independently from semantics. Consider this sequence:

The girl eats the apple with a smile.
How does the girl eat the apple? – With a smile

Suppose we tried to write a computer program which could engage in conversations like this. The simplest approach seems to be to use pattern-matching. We might look for a sentence with the pattern:

The <noun1> <verb> *s the* <noun2> *with a* <noun3> .

Then if we had a question about this sentence which had the pattern:

How does the <noun1> <verb> *the* <noun2>?

the program could answer:

With a <noun3>.

Thus given:

The man closes the door with a bang.

the program could cope with the sequence:

How does the man close the door? – With a bang.

But now suppose we try the sentence:

The girl eats the apple with a bruise.
How does the girl eat the apple? – With a bruise. **wrong!!**

The problem is that although on the surface the two sentences *the girl eats the apple with a smile* and *the girl eats the apple with a bruise* are similar, their underlying syntax

is quite different: *with a smile* qualifies *eats* in the first sentence whereas *with a bruise* qualifies *the apple* in the second sentence. We can try to show this by using parentheses:

[The girl] [eats (the apple) (with a smile)]
[The girl] [eats (the apple (with a bruise))]

Unfortunately, to work this out our program will need to know that, for example, apples can have bruises but not smiles.

Now consider:

The girl eats the hamburger with relish.

This is ambiguous: it may mean that the hamburger has relish on it or it may mean that the girl is eating the hamburger with enthusiasm. Programming languages are deliberately constructed so that their syntax is unambiguous, but NLs are not. Thus **processing the grammar of natural languages depends on simultaneous processing of meaning.**

1.5 Meaning

Meaning can be subdivided in many ways. A simple division is between semantics, referring to the meaning of words and thus the meaning of sentences formed by those words and pragmatics, referring to the meanings intended by the originator of the words. Note the meaning of sentences depends not only on the meaning of the words, but also on rules about the meaning of word combinations, word orders, etc. For example, in the English sentence the boy sees the girl, we know that the boy does the seeing and the girl is seen because in English the normal word order is subject – verb – object.

A striking feature of NLs is that many words and sentences have more than one meaning (i.e. are semantically ambiguous), and which meaning is correct depends on the context. This problem arises at several levels.

There are problems at the level of individual words. Consider:

The man went over to the bank.

What kind of 'bank'? A river bank or a source of money? Here we have two distinct English words with the same spelling/pronunciation.

However, there are more subtle problems at the word level. Consider:

Mary loves Bill.
Mary loves chips.

The word loves in these two sentences does not have precisely the same meaning (and might need translating by different words in other languages). This is especially clear if you try changing *Bill* to *chips* in:

Mary loved Bill – that's why she killed herself.

Metaphorical use of words also causes problems. Consider:

Water runs down the hill.
The river runs down the hill.
The road runs down the hill.

In the first sentence, *runs* implies movement; in the third it does not. What about the second? Other languages may not allow the same word to be used in all three senses. The literal/metaphorical boundary tends to shift with time and usage.

There are also problems at the sentence level. The meaning of a sentence is not just the meaning of the words of course, it also involves knowledge of the rules governing the meaning of word combinations, orderings, etc. in the NL. However, even given this knowledge, idioms cause problems:

He really put his foot in it that time.

The classic example from MT is the system that translated:

The spirit is willing but the flesh is weak.

into Russian, and then translated the Russian back into:

The vodka is good but the meat is rotten.

At the passage level endless problems arise. 'Anaphora' (e.g. substituting less meaningful words such as pronouns for nouns) is a common feature of NLs. Suppose we read the sentence:

London had snow yesterday.

and then read **one** of the following sentences, all of which are sensible continuations. What is the *it*?

It also had fog. (It = London)
It fell to a depth of 1 metre. (It = the snow)
It will continue cold today. (It = ?the weather)

Handling pronouns is a difficult task, and seems to require considerable world knowledge and inference. (Even in MT where 'understanding' the text is apparently not an issue, inferring the original of pronouns can be important, because it may be necessary to reflect gender; e.g. if translating the above into French, *it* may become either *il* or *elle*.)

Conclusion : The meaning of natural language cannot be established from the meaning of the words plus rules for determining meaning from word combinations and orderings; inference and world knowledge are needed.

If all this weren't enough, we need to consider the (inferred) intention of the producer of the NL being analysed. It may be thought that this is covered by semantics, but a few examples soon show otherwise. Consider the question:

Can you tell me the time?

A syntactically and semantically perfectly correct response is *Yes*! Pragmatically, this is wrong (assuming the question was asked in a 'normal' context by a native speaker of English), since the intention behind the question is that the respondent should **tell** the questioner the time. How can a program be written which deals correctly with the difference between

Can you swim?

and

Can you pass me the life belt?
The difference between

Pass me the salt.
Pass me the salt, please.

Can you pass me the salt?

Could you pass me the salt?

is essentially in levels of politeness, rather than in any other aspect. This issue is particularly important in MT, where literal (i.e. syntactically and semantically correct) translations may not be pragmatically so. **Conclusion: The surface meaning of a sentence is not necessarily the meaning intended by the producer.**

Chapter 2

Of probabilities, polarities and peculiarities

Automatic extraction of human opinions from Web documents has been receiving increasing interest. The vast amount of user-genererated media available on the Web can be of immense use once we have a system that could tag them under certain classes. This is popularly known as **opinion extraction** or **sentiment mining**. Generally speaking, it is the ability to identify polar expressions in unstructured data. The main tasks that we have at hand are the following:

- mining product features that have been commented on
- determining whether the comment/opinion on each product feature in each sentence is *positive*, *negative* or *neutral*
- summarizing the results

In this chapter, we'll attempt to tackle the problem of classifying a document as positive, negative or neutral. That's because the other two are comparitively much easier than this. As a product can only have a finite set of features, identifying which one of these is being talked about in a given piece of text is a word matching task, only in a slightly complicated format. (As a feature could be a set of words, or even worse, a set of signal words implying a feature). Summarization, basically, is presenting the mixed opinion which is retained after seeing a large collection of documents. This can easily be achieved once the other two problems get tackled for individual posts.

2.1 Text Classification Algorithms

The goal of classification is to build a set of models that can correctly predict the class of the different objects. The input to these methods is a set of objects (i.e., training data), the classes which these objects belong to (i.e., dependent variables), and a set of variables describing different characteristics of the objects (i.e., independent variables). Once such a predictive model is built, it can be used to predict the class of the objects for which class information is not known a priori. The key advantage of supervised learning methods over unsupervised methods (for example, clustering) is that by having an explicit knowledge of the classes the different objects belong to, these algorithms can perform an effective feature selection if that leads to better prediction accuracy.

2.1.1 Naive Bayesian Classifier

Naive Bayesian (NB) algorithm has been widely used for document classification, and shown to produce very good performance. The basic idea is to use the joint probabilities of words and categories to estimate the probabilities of categories given a document. NB algorithm computes the posterior probability that the document belongs to different classes and assigns it to the class with the highest posterior probability. The posterior probability of class is computed using Bayes rule and the testing sample is assigned to the class with the highest posterior probability. The naive part of NB algorithm is the assumption of word independence that the conditional probability of a word given a category is assumed to be independent from the conditional probabilities of other words given that category. There are two versions of NB algorithm. One is the multi-variate Bernoulli event model that only takes into account the presence or absence of a particular term, so it doesn't capture the number of occurrence of each word. The other model is the multinomial model that captures the word frequency information in documents. The details have not been provided as it's available in any standard Data Mining textbook.

2.1.2 Support Vector Machine

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. They belong to a family of generalized linear classifiers.

A special property of SVMs is that they simultaneously minimize the empirical classification error and maximize the geometric margin. Viewing the input data as two sets of vectors in an n-dimensional space, an SVM will construct a separating hyperplane in that space, one which maximizes the "margin" between the two data sets. To calculate the margin, we construct two parallel hyperplanes, one on each side of the separating one, which are "pushed up against" the two data sets. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the neighboring datapoints of both classes. The hope is that, the larger the margin or distance between these parallel hyperplanes, the better the generalization error of the classifier will be. More specifically, we view a data point as a p-dimensional vector (a list of p numbers), and we want to know whether we can separate them with a (p-1)-dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. However, we are additionally interested in finding out if we can achieve maximum separation (margin) between the two classes. By this we mean that we pick the hyperplane so that the distance from the hyperplane to the nearest data point is maximized. That is to say that the nearest distance between a point in one separated hyperplane and a point in the other separated hyperplane is maximized. Now, if such a hyperplane exists, it is clearly of interest and is known as the maximum-margin hyperplane and such a linear classifier is known as a maximum margin classifier. Mathematical details have been omitted as it's available in standard texts.

2.1.3 Observations

Despite being able to get a decent accuracy in two-class classification, it was often the case that we could not extend it to three-class, while maintaining the accuracies. While the distinction between *positive* and *negative* was often clear, the *positive-neutral* and the *negative-neutral* classification was turning out to be difficult. While observing the features closely, we reached the conclusion that we have to modify our feature extraction algorithm. Till now, we were considering all tokens as features (subject to morphing, after removing stop-words). This, as we figured out, gives a lot of those words as features which lack classifying power. Thus, **feature seclection** becomes quite crucial for better classification.

2.2 Feature Selection

There potential benefits of feature selection are:

- facilitates data visualization and data understanding
- reduces the measurement and storage requirements
- reduces training and utilization times
- defies the curse of dimensionality to improve prediction performance

2.2.1 Sensitivity Analysis

Sensitivity analysis is used to determine how sensitive a model is to changes in the value of the parameters of the model and to changes in the structure of the model.

Ideally, it would be great to have only those features in the feature vector which are peculiar to one class. One way is to select only those tokens which strongly express some sentiment like love, hate etc. But then what about sentences like

I hate the fact that I don't have PlayStation3.

This sentence is strongly positive about PlayStation3, despite 'hate' being used in the sentence. That means something as simple as this may not work. What we tried was to look at the Information Gain that we get by including a feature in our feature vector, compared to others. In simple words, if a token is able to give us valuable information (needed for classification), then only it'll be included in the list. Such an algorithm is inherently capable of discarding features with strong sentiment if they are incapable of classifying a document. This decrases the density of articles near the classifier margins and hence reduces the probability of a document being misclassified.

Chapter 3

For Better Viewing, Highlight

As we were discussing, the user-generated media available on the Web is vast. Not to forget, it's expanding. Would it not be nice for a reader to be able to get an idea of the matter without actually going through each and every sentence? If we look at the documents closely, we can observe that people try to express opinions in a certain number of ways. In other words, when an author is writing a document, there are a set of sentences that he writes just to make the article look like a story. These sentences tend to have a different structure compared to those sentences that take the story forward. In case of a blog or a review, these distinctions become even clearer.

3.1 An Intuitive Idea

A natural approach is to observe sentence structures to be able to predict the importance of a sentence is an article. The idea is to read an article and to be able to decide the importance of the current sentence from what's already been read. The tool we came up with (**SentiMiner**) is rule based. The rules, however, are carefully written with placeholders to accomodate author's creativity. The results were quite impressive.

3.1.1 SentiMiner

Let us look at an examples. Let us consider the following article:

Acme Co.'s entry into the music sales space has been greeted with a less than enthusiastic response *by those who have used the service. While* ***it would be foolish to imagine that it won't get better****,* ***Acme Co. has a bigger problem*** *due to it's choice of audio file format than Apple does.*

Right now ***Acme Co's service doesn't support the most popular digital audio player*** *on the market. The iPod.*

The iPod sold out completely this Christmas and has generated forests of newsprint at even the whiff of mini-iPods. (A budget line of players such reports say will make a showing at Macworld next week.) For Acme Co. to beat the iTunes Music Store, something has to come along and beat the iPod. ***The iTMS itself is pretty worthless*** *as a business,* ***it will never turn a healthy profit*** *and only serves as one more reason to buy the market leading iPod. With* ***the music industry taking such a large cut of the music tracks retail price*** *there's very little money to be made on the music and* ***a whole lot to be made on the player.***

With Macworld so close ***it would be stupid to comment on...***

The highlighting enables the reader to go through the article very quickly, without missing out on any important information. Let us see another one.

A chilly morning breeze greeted me as I walked out of the lobby of my building*. I smiled to myself, thinking ahead to February.* ***A much colder environment awaited us there****. My mind wandered.* ***I couldn't remember the last time I jammed*** *or was involved in* ***any music related activity with friends****. Probably Nostalgia 2006. Too long a gap. Talk about losing my way.*

ND had been breaking her head for song(s) ***she could perform in a small society gathering****. Having come across very few choices, she'd spoken to me about it.* ***I wasn't much help either*** *but somewhere along the way, we stumbled upon "Desperado" by the Eagles (her suggestion),* ***which in my opinion turned out to be quite a scorcher****. For starters, I had not heard the song too often. Yesterday I got home and I ripped the song from the CD. Then I heard the track on (****a trusted****) Winamp repeat mode and fixated myself on it for a while.*

When we met in college, KP & MG had joined in the fun. MG got his guitar along while ***KP saw the entire plan as a great opportunity to start things*** *off on what he'd been looking forward to for sometime now.* ***We started off a little awkward****. I relied on ND's familiarity with the song's tune. It helped me pinpoint the chord timing.* ***I'm very finicky about that****, no doubt (MJ is grinning ear to ear at this point). I have to get it spot on or I'm not going to move ahead. Meanwhile,* ***the two other musicians were busy discussing various chord formations*** *and* ***exercising their fingers like never before****.*

We got rolling in half an hour and ND picked up the pace. ***She sings well****. Don't know why* ***that stage fright syndrome rings in for all of us****. Post 26th, things should change. I have another 'session' tomorrow. Punctuality? Yea. 9 a.m. Give it all I've got.*

This idea was also extended to summarize pieces of text without losing any valuable information on the sentiment expressed. The higlighted regions of text can be used as one. The only problem being, there are times when only parts of a sentence gets highlighted. When a reader is given the whole post, he can see near the highlighted regions to get a sense of what is being expressed. While if the extract alone is shown, it becomes quite important that the chunks of text seem meaningful in themselves and related to other chunks. It was not too difficult to tweak the same to match the requirements. The results were equally good.

Chapter 4

"Cut" the Crap

In the last chapter, we looked at a way to summarize a piece of text while retaining all the useful information. That, however, was an intuitive approach to the problem. Here we'll look at summarization from a completely different perspective. The idea comes from the work of Pang and Lee [1].

4.1 An Algorithmic Approach

The idea to classify sentences in a document as either *subjective* or *objective*. The set of subjective sentences is considered as a summary of the document. Sentiment Mining using just the summary gives better accuracy. This decreases the training and testing time as well, as about 40% of the sentences get discarded in the summarization process.

To understand this concept broadly, let us consider a movie review. There is difference between sentences that talk *about* the movie and those that describe what's *in* the movie. A bad character in the movie can be talked about as *really bad and scary*, which is a good thing. Similarly, although the sentence '*The protagonist tries to protect her good name*' contains the word '*good*', it tells us nothing about the author's opinion and in fact could well be embedded in a negative movie review. To get a better idea of this, let's look at some subjective and objective sentences taken from movie reviews.

- **Subjective** - *It's a good thing that woolly mammoths are extinct, because this movie will have every kid in the schoolyard wishing for their very own.*

- **Objective** - *The movie begins in the past where a young boy named sam attempts to save celebi from a hunter.*
- **Subjective** - *The only young people who possibly will enjoy it are infants ... who might be distracted by the movie's quick movements and sounds.*
- **Objective** - *Television made him famous, but his biggest hits happened off screen.*
- **Subjective** - *Anyone who can count to five (the film's target market ?) can see where this dumbed-down concoction is going.*
- **Objective** - *Jordan is a mom who is on a life long search for true faith as she tries to protect her only child from what she believes is injustice.*
- **Subjective** - *The movie is pretty funny now and then without in any way demeaning its subjects.*
- **Objective** - *Consequently, what begins as an enthusiastic road trip is soon plagued with mysterious roadside obstacles that threaten to prevent the boys from ever making it to the competition.*

4.1.1 Subjective-Objective Classification

As with document-level polarity classification, the subjectivity detection can be performed on individual sentences by applying a standard classification algorithm on each sentence in isolation. However, modeling proximity relationships between sentences would enable us to leverage *coherence*: text spans occurring near each other (within discourse boundaries) may share the same subjectivity status, other things being equal (Wiebe, 1994).

There are two kinds of scores that comes into the picture. First is an *individual* score of a sentence, for subjective as well as objective class. This, in some sense, models how happy will a sentence s be, if it's classified as subjective (similarly, for objective). These values can be the probability of a sentence falling to each class, based on supervised learning methods. The other score is an *association* score, quantifying the happiness of two sentences when both are given the same class. The idea is to minimize the net '*unhappiness*' of the system. This problem beautifully translates to a graph-theoretic problem of finding *min-cut*. Details of the same have been omitted here.

This has been further extended by Agarwal and Bhattacharyya in [2], which helped a lot in improving the accuracy of polarity classifiers.

Chapter 5

Tantalizing Titles

With the increasing volume of content available to us through the World Wide Web, it becomes vital for a system to be able to detect duplicates. Detecting duplicate and near duplicate documents is an area of research in itself and there are many ways to do that. But what we are trying to attack here is slightly different. We are trying to exploit the information possessed by the title of the posts and not the posts themselves. Although such an approach is unlikely to give great accuracy figures, what we are currently interested in is faster processing.

5.1 Title Duplicacy

Let us consider the following set of titles:

Subprime crisis hits ICICI Bank
ICICI Bank is hit by subprime crisis

Tata Sky to restore ESPN Channels in basic package
ESPN Channels now in the basic package of Tata Sky
Basic package of Tata Sky to include ESPN

The article under these titles are quite likely to be the same. More so, when we know that they were published on the same day. The idea is to club them together because:

- it saves us from the load of processing those extra (similar) articles
- it enables us to see how a particular news spreads on the Web, and when/how it starts affecting end users of a product/service

It's not very difficult to see that such titles overlap. If we could quantify the overlap of two titles then it would be easier to group them. The approach can be as simple as transforming all the words to their respective base words (and removing the stop-words) to see how many words are actually the same. While this is quite a simple rule, it works more often that we would expect it to work. There are obvious drawbacks such as:

- it would not work for small titles
  for example, two different people who dislike a product P can write different articles, both titled '*P sucks*'.
- consider the following titles:
  Microsoft bids to buy Yahoo
  Microsoft raises bids to buy Yahoo

  Although these two titles differ by just a word, it would be inappropriate to group them together.

It may look complicated on the surface to catch such a behaviour, but it's not quite so. If we look closely at what happened in the last example, we would see that although there's only one word which is extra in the second title, that extra word is a verb. This clearly describes an action which *Microsoft* is doing (precisely, *raising* it's bid), and is new (the earlier title does not talk about any such action). Now it becomes obvious that these cannot be the same. The second article talks about something new, a new action being performed.

Some similar observations have enabled us in designing a tool which can predict the possibility of two articles being 'similar' by looking at their titles.

Chapter 6

What's in a name?

Detection of Named Entities is another challenging task that needs to be tackled. While a clear advantage of this would be the fact that we'll be able to eliminate named entities from the feature list, we would also be able to predict a lot about the author. Moreover, if a product P has services is regions A and B, from a post which talks negatively about P we want the ability to say that the problematic region is A (say) if the author belongs to the region A. It's highly unlikely that such facts will be stated in the article. One approach is to look for signals, a set of tokens that are much more likely to be present in a post specific to certain region. If we can quantify the strength of these signals, it would be possible to map posts to different geographies.

6.1 Named Entity Recognition

A Named Entity (NE) is a word, or sequence of words that can be classified as a name of a person, organization, location, date, time, percentage or quantity. Named entities can be valuable in several natural language applications. For instance, automatic text summarization systems can be enriched by using NEs, as they provide important cues for identifying relevant segments in text. Other uses of NE taggers are in the fields of information retrieval (i.e. more accurate Internet search engines), information extraction, automatic speech recognition, question answering and machine translation.

While for a human the task of identifying in a text which parts correspond to NEs can be trivial, the same cannot be said about designing programs capable of classifying

named entities with human-level performance. This is a difficult goal to achieve due to a common problem of all natural language processing tasks: ambiguity. Another inconvenience is that documents are not uniform, their writing style, as well as the vocabulary can change dramatically from one document to another. Thus, there has been a considerable amount of work that aims to improve the performance of NE taggers.

While Machine Leaning approach can be applied to solve such a problem, this requires a great deal of supervision. There have been recent advances in this area and successful efforts have been made to reduce the size of annotated data that's needed for building a large scale system for NE Recognition.

6.1.1 Geography Identification

What we tried for Geography Identification was quite simple. The problem at hand was to classify articles as *Indian / Non-Indian*. We compiled a long list of names, surnames, places, foods, festivals, languages, rivers, castes to obtain different types of Indian signals that a document can contain. To start with, we assigned equal weightage to all the signals. The values were later tweaked for improving results. The tool performs well, and can easily be scaled to do multi-class classification.

Chapter 7

Conclusion

NLs are **massively locally ambiguous** at every level (speech, grammar, meaning). Yet in normal use, NL is effective and rarely ambiguous. To resolve local ambiguity, humans employ not only a detailed knowledge of the language itself – its sounds, rules about sound combinations, its grammar and lexicon together with word meanings and meanings derived from word combinations and orderings – but also:

- A large and detailed knowledge of the world.
- The ability to follow a 'story', by connecting up sentences to form a continuous whole, inferring missing parts.
- The ability to infer what a speaker meant, even if he/she did not actually say it.

It is these factors that make NLs so difficult to process by computer – but therefore so fascinating to study.

Bibliography

[1] Bo Pang and Lillian Lee. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. *ACL 2004*, 271–278, 2004. URL www.cs.cornell.edu/home/llee/papers/cutsent.pdf.

[2] Alekh Agarwal and Pushpak Bhattacharyya. Sentiment Analysis: A New Approach for Effective Use of Linguistic Knowledge and Exploiting Similarities in a Set of Documents to be Classified. *International Conference on Natural Language Processing (ICON 05)*, 2005. URL www.cse.iitb.ac.in/~pb/papers/alekh-icdm-new-14jun05.pdf.