**Paper Summaries**

1. SemEval 2013 – Task 2: Sentiment Analysis in Twitter

This paper involved 2 main tasks:

1. Expression-level subtask
2. Message-level subtask

Important things mentioned

* Data from twitter may contain creative spelling, misspellings, slang, new words, URLs and specific terminology and abbreviations related to the topic being spoken about
* Main aim of promoting research of how sentiment is conveyed in tweets and SMSs

Subtask A: Contextual

* Given a message containing a marked instance of a word. Determine if that instance is positive, negative or neural.

Subtask B: Message

* Given a message, determine if it is positive, negative or neural.

Constrained – could only use provided data for training

Unconstrained – could use any additional data to train the system.

Data collection – Twitter API

Tweets which had no sentiment were removed

SentiWordNet – used for filtering

Results for subtask A were generally better than subtask B – less ambiguity at phrase level and a message can contain both positive and negative phrases together.

Twitter messages can be noisy – some participants performed prepossessing.

Deep Learning important for tweet sentiment analysis Semeval 2016 Task 10.

Research quantification (more than positive or negative) - not just classification(positive, negative or neutral) Semeval 2015 Task 10

Semeval 2014 Task 9 - It is good to identify sarcastic tweets before for filtering, #sarcasm tag?

SentiWordNet - identify tweets that contain sentiment bearing words.

WordNet.

Tweet Sentiment: From Classification to

Quantification

Wei Gao and Fabrizio Sebastiani - Papaer about different algortihms, some things were a bit difficult

**Ntlk – python**

**Datatset generic or specific?]**

**Mandeley**

**Chapter5, 6, 7, 9:**

**Chapter 5 Summary – Logistic Regression**

* Logistic Regression works better on large datasets
* Naïve Bayes works better on vert small datasets
* If it’s too high, the learner will take steps that are too large, overshooting the minimum of the loss function. If it’s too low, the learner will take steps that are too small, and take too long to get to the minimum. It is common to begin the learning rate at a higher value, and then slowly decrease it

**Chapter 6: Vector Semantics and Embeddings**

Finding similarities between words.

Create a matrix between words and documents. Page 8

Sometimes we can find similarities by taking a -+4 window and then create a matrix of words and words which are next to each other rather than a matrix between words and documents. (**word-word matrix or term-context matrix**)

Two main approaches:

1. Cosine for measuring similarity (Page 11)
2. TD-IDF Algorithm

This weighting is the way for weighting co-occurrence matrices in information retrievak.

Two factors:

1. **Term frequency** - Take the count of the word in the document using the log10 and adding one to avoid log of 0. This is the amount of times a term occurs in a document
2. **Document frequency** – This is the amount of documents the term occurs in

The inverse document frequency = log10 [N (number of documents) / df]. This is usually squashed using a log10 due to the large number of couments.

Tf-idf weighted value = tf x idf

If a common word such as good occurs in all documents, idf would be 0 since log 10 of 1 is 0 and hence tf-idf weighted value is 0. Therefore, that common word is simply ignored.

This can be used to see if two documents are similar

An alternative for TF-IDF is PPMI (Positive Pointwise Mutual Information)

Sparse vs Dense word vectors reasons:

1. dense vectors may be more successfully included as features in machine learning systems
2. they contain fewer parameters than sparse vectors of explicit counts, dense vectors may generalize better and help avoid overfitting
3. dense vectors may do a better job of capturing synonymy than sparse vectors.

Word2vec simpler than neural network:

1. Simpler: binary classification instead of word prediction
2. Simpler architecture: training a logistic regression rather than a multi-layer neural network

Skip-Gram with negative sampling = method for dense, short word vectors.

The probability is based on applying the logistic (sigmoid) function to the dot product of the embeddings of the target word with each context word.

When learning skip-gram embeddings, we need more negative samples than positive samples.

**Note that the skip-gram model thus actually learns two separate embeddings for each word** w: **the target embedding** t and the **context embedding** c.

Two words have first-order co-occurrence **first-order co-occurrence** (sometimes called syntagmatic association) if they are typically nearby each other. Thus wrote is a first-order associate of book or poem. Two words have **second-order co-occurrence** (sometimes called paradigmatic association) if they have similar second-order co-occurrence neighbors. Thus wrote is a second-order associate of words like said or remarked.