Domain Specific Sentiment Lexicons Induced from Labeled Documents

Ulysse Marquis
Politecnico di Torino
s293793@studenti.polito.it

Fabio Tatti
Politecnico di Torino
s282383@studenti.polito.it

Andrea Silvi
Politecnico di Torino
andrea.silvi@studenti.polito.it

***Abstract*—In this report we expand on the concept of Domain Specific Sentiment Lexicons by trying different methods to detect negated words and explore how lexicons can vary among different time periods and different internet communities. You can find the implementation code of this paper at *github.com/marquis-silvi-tatti/DomainSpecificLexicons*.**

I. Problem Statement

Sentiment analysis is one of the more common natural language processing tasks, seldom framed as a text classification task, where one wants to assign a positive or negative label to open-ended text. In order to tackle this task, one can exploit sentiment lexicons that can be run out-of-the-box without the need of labeled data.

A sentiment lexicon is a mapping between a word *w* and a polarity score which usually lies in the range $[-1, 1]$. Depending on the sentiment distribution, one can assess relative sentiments differently. In practice, vocabularies include so many neutral words that the score distribution's mean is very close to zero. Then, one can leverage these scores in order to assess the overall sentiment of a document, by averaging the scores of each word in the document and obtaining an overall sentiment value.

One perennial problem with sentiment lexicons is that they are usually based on domain-independent notions of sentiment polarity. In reality, sentiment lexicons are never static and easily change between different domains, but they can also change in time: *awful* for example used to refer to something "worthy of awe", often being used with a positive connotation in expressions such as "the *awful* majesty of God", while now it holds a very negative sentiment polarity.

In order to obtain a domain specific lexicon, this method first process a domain-specific corpus $\mathcal{D}$ composed of n documents $\{x_i\}_{i=1}^{n} \in \mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is the set of all possible documents generated with the vocabulary $\mathcal{V}_\mathcal{D}$ and $\mathcal{Y} \in [-1, 1]$. Then it learns a mapping $\psi : \mathcal{V}_\mathcal{D} \rightarrow [-1, 1]$ and, given a vocabulary of words with relative word embeddings $\mathcal{G} = \{w : v(w)\}_{i=1}^{N}$, it infers a score $\psi(w)$ for each word $w \in \mathcal{G}$.

II. Methodology

We follow the two-step approach of [1]: first we leverage domain-specific labeled documents to train a linear predictor to induce the score of the seed data, which is comprised of the more frequent words in the input corpus. Then, we train a deep regression network on this seed data to learn a mapping

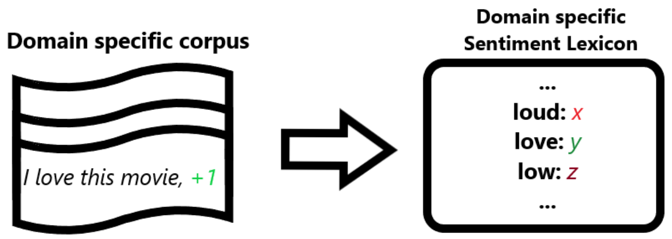

Fig. 1: **Sentiment Lexicon generation schema.** From a set of labeled documents, this method aims to generate a lexicon consisting of a sentiment score for each word in it.

between words and scores in order to extend the scores also to a larger vocabulary, exploiting word vector representations.

A. Document pre-processing

Given a domain specific corpus $\mathcal{D}$, we lowercase all documents and we define the set of features as $\mathcal{F} = \mathcal{V}_\mathcal{D} \cup \{\bar{w_j} \mid w_j \in \mathcal{V}_\mathcal{D}\}$, where $\bar{w_j}$ is the negated version of word w_j. We then tokenize each document $x \in \mathcal{D}$ by word and map it to the set of features $\mathcal{F}$, thus obtaining a bag of words model for each of them.

We investigate different methods for detecting negations in each review x:

- as in [1], we simply negate a word w_j to $\bar{w_j}$ if it is proceded by a *not*;
- whenever we find a *not*, we negate each word w_j that comes after it to $\bar{w_j}$, until the sentence ends;
- we leverage a dependency parser module to find negation relations between words, and we negate words that present this relationship.

We also report a simple example of the different methods in Table I.

1) Simple negations: we call *simple* the negation detection method used in [1], since it is very fast and can process huge amounts of documents in a short time, but can easily incur in errors, since many times the word a negation refers to does not appear directly after it.

2) Whole negations: we then implement the common negation detection method of negating all words that come after a negation. Seldom a negation does not negate a single word in a phrase, but rather switches the sentiment of the whole subsequent phrase. Consequently, negating all the words after a negation tries to emulate this phenomenon.

Type of negation / Input phrase	*I do not really like this movie.*
simple	I do **NEG**really like this movie.
whole	I do **NEG**really **NEG**like **NEG**this **NEG**movie.
complex	I do really **NEG**like this movie.

TABLE I: **Impact of different type of negation detection methods.** We report the outputs of a single toy phrase after being processed through the different methods presented in Section II-A. Note that for the *complex* negation method, the dependency parser links the word *not* with the word *like* through a **negation** relation, thus resulting in the output of the last line.

3) Complex negations: finally, we come up with a more elaborate way to detect negations, by leveraging a syntactic dependency parser. This module ϕ, given a sentence y composed of n words $w_i (i = 1, ..., n)$, builds a graph $\phi(y) = \mathcal{G}_y$ where each word w_j is a node and an edge (i, j) exists if w_i and w_j are found to be connected by a syntactic relation in the original phrase y. We then negate words that are found by ϕ to be the tail of an edge of type *negation*.

B. Seed data induction

After mapping each document x to the set of features $\mathcal{F}$ with the methods described above, we obtain a document-term matrix X composed of n documents $\bar{x}$ in a bag-of-words format that we use to train a linear model $f(\bar{x}) = \omega^{\intercal}\bar{x}$.

Finally, for each word $w_j \in \mathcal{V}_{\mathcal{D}}$, we filter out words that do not appear enough in the corpus, and we assign to the remaining ones the score obtained from the linear predictor ω_j learned by f. We then obtain the training data, that we also call *seed* data, as

$$T = \{(\mathbf{v}_{w_j}, \omega_j) \mid freq(w_j, \mathcal{D}) \geqslant f_{min}\}$$

where $\mathbf{v}_{w_j}$ is the word vector of word w_j, $freq(w_j, \mathcal{D})$ is the number of times the word w_j appears in the corpus $\mathcal{D}$ and f_{min} is the minimum frequency that, as in [1], is kept at 500 for all of our experiments.

We discard negated features as in [1], since their frequency is too low to provide a reliable complementary signal, except we also try to combine the coefficients of a word w_j and of its negated self $\bar{w}_j$, which we call ω_j and $\overline{\omega_j}$, when we use the *whole* negation identification method, since we hypothesise that because this method negates many more words than the other ones means that negated features can also carry a reliable signal. More specifically, for each word $w_j \mid freq(w_j, \mathcal{D}) \geqslant f_{min}$ we obtain the score for word w_j as

$$\widetilde{\omega_j} = \begin{cases} \frac{\omega_j + \overline{\omega_j}}{2}, & \text{if } freq(\bar{w}_j, \mathcal{D}) \geqslant f_{min} \\ \omega_j, & \text{otherwise} \end{cases} \tag{1}$$

where $\overline{\omega_j}$ is the linear predictor corresponding to the feature of the negated word of w_j and $freq(\bar{w}_j, \mathcal{D})$ is the number of times the negated word $\bar{w}_j$ appears in the corpus $\mathcal{D}$, after being processed with the *whole* negation detection method.

C. Neural Word-Vector based Label Expansion

Using T as training data, we finally train a model $\psi(\mathbf{v}_w) \in \mathbb{R}$ to predict real-valued sentiment polarity scores. Since word vectors contain semantic information about the words, including their sentiment [2], they help overcome the problem that we do not have sentiment polarity information for the majority of the words after the seed data generation.

Training the model ψ on the word vectors of our seed dataset T means that we can use the same model ψ to predict the sentiment score of the other words that do not appear in T and still expect their score to be representative and domain specific: the word vector finds where the unseen word w_i lies in the word vector feature space and this means that $\phi(\mathbf{v}_{w_i})$ is going to return a score close to those of close words to w_i that are actually present in T. Moreover, since ϕ is trained on domain specific scores from T, the score for an unseen word w_i is going also to be domain specific: if for example we are training ϕ on a seed dataset T obtained from a corpus of music reviews, and in T we find the word *hot*, which has a positive connotation in the music domain, but we do not find *fire*, the word embedding of *fire* plus the positive label that we get in T for *hot* means that after the prediction step also *fire* will have a positive connotation. Obviously, the same example on a different domain like electronics reviews will show that both words are going to have negative sentiment polarity scores (one generally does not associate happy feelings with her laptop being *hot* or on *fire*).

We use T to train the same network ϕ as [1], and after training we use the same model ϕ to extend the labels to the whole vocabulary $\mathcal{G}$ we are interested to have as our lexicon.

III. Experiments

In this section we report our different experiments and our findings from them: in Section III-A we build 12 different lexicons using corpora coming from different domains and different negation detection methods and exploit them in order to do unsupervised review sentiment classification. Then in Section III-B we generate 3 different lexicons on the same dataset but on reviews written in different years, in order to explore if and how much some words can change meaning in the span of a couple decades. Finally, in Section III-C we build 2 lexicons on contrasting Reddit communities in order to see in what way the correlation between the use of a word and the amount of upvotes minus downvotes a comment with that word in it gets changes between opposite communities.

In every experiment we train the deep regression network for $n = 100$ epochs, using Google Colab GPUs as our hardware. We use Adam optimization with starting learning rate of 0.001, batch size of 32, dropout rate of 0.2 and early stopping.

We use SpaCy[1] dependency parser in order to obtain the dependency tree to detect negations in method II-A3 and we

[1]https://spacy.io/usage/linguistic-features

Training dataset	**Movies**		**Pet supplies**		**Videogames**	
Type of negation / Test dataset	**imdb**	**gamestop**	**imdb**	**gamestop**	**imdb**	**gamestop**
simple	0.869	0.849	0.770	**0.830**	0.825	0.889
whole w/o negated features	0.870	0.839	0.771	0.827	**0.832**	0.880
"whole" w/ negated features	0.864	0.841	0.769	0.813	0.825	0.879
complex	**0.876**	**0.857**	**0.776**	0.824	0.831	**0.891**

TABLE II: Accuracy results for different combinations of input corpus (main columns) and negation detection methods. The results in bold are the the best performing ones.

use Cornell's Conversational Analysis Toolkit (ConvoKit)[2] to obtain and manipulate subreddits data.

A. Unsupervised Reviews Sentiment Classification

We use as our input corpus 3 different datasets containing Amazon reviews that span from 1996 to 2018 and can be found on the Internet [3]:

- the first one contains 3.41 million reviews of movies and tv series;
- the second one contains 2.57 million reviews of videogames;
- the third one contains 2.10 million reviews of general pet supplies.

We consider reviews that have a score greater than 3.0 as positive, and as negative those with a score lower than 3.0. Reviews with a score of 3.0 are considered as neutral and subsequently discarded.

We follow the approach presented in Section II building a different lexicon for each input corpus and for each different negation detection method described in II-A. For the *whole* negation method we build one lexicon without utilizing the predictors of the negated seed data words and another one that leverages Equation (1) to find the scores of the seed data. We always use as minimum frequency $f_{min} = 500$ as in [1].

Consequently we obtain 12 different lexicons based on these three corpora and the four different negation detection techniques, using as word vectors GloVe CommonCrawl embeddings [3] and also using the same GloVe words as the vocabulary of our lexicons. We use these to predict unseen reviews sentiments from two different datasets:

- the IMDB movie reviews dataset compiled in [4], comprising of 25000 positive and 25000 negative reviews;
- a dataset[4] comprising of 433 positive an 433 negative scraped reviews of GameStop items.

We train our lexicons on different domains in order to understand how much the similarity or dissimilarity between training and testing domains affects the results of our predictions. In order to predict the sentiment of a review, we average the scores of the words of the review for which we have a score in the lexicon, and put at 0 the score of the others, more specifically $f(x) = \frac{1}{|x|}\sum_{i=0}^{|x|}\phi(\mathbf{v}_{x_i})$, where x is the unseen review composed of $|x|$ words x_i, and $\phi(\mathbf{v}_{x_i})$ is the score of word x_i in our lexicon. As in [1], we find that using as threshold for deciding if a reviews is positive or negative the average of the prediction scores gives us the best results.

[2]https://convokit.cornell.edu/documentation/index.html#cornell-conversational-analysis-toolkit-convokit-documentation

[3]https://nijianmo.github.io/amazon/index.html

[4]https://drive.google.com/file/d/1GWQatoemfcy2kiupmmKrt43fbraOYIav/view?usp=sharing/

Negation method	**seconds per 1k documents**
simple	$6.4\ 10^{-3}$
whole	0.2
complex	19

TABLE III: The average amount of time it takes to process 1000 documents using the different negation detection methods portrayed in Section II-A.

Group of years	**Positive**	**Negative**
1995 - 2004	23350	2425
2009 - 2013	21169	4606
2016 - 2018	23720	2055

TABLE IV: The number of positive and negative words for each lexicon generated from reviews of different group of years, considering words with a score $S < -0.20$ or $S > 0.20$ as non-neutral.

The results are reported in Table II. As we can see, for most runs the lexicons obtained by parsing the input corpus using the negation detection method that leverages the dependency parser obtain the best results. Moreover, as we expect the lexicons trained on the video games domain are better than the others at predicting the sentiment of GameStop reviews, and the same happens with the lexicon trained on the movies domain when predicting IMDB movies reviews. Also, as expected the lexicons trained on the pet supplies domain, which surely tends to be quite different from the domains of digital media like video games or movies, obtain the worst results.

We also compare the time it takes for each different negation method to process 1000 reviews in order to be as fair as possible. The results are reported in Table III. As we can see, we obtain a small gain in terms of accuracy from the *complex* negation detection method at the expense of a much higher input corpus preprocessing time. This huge increase in computation time makes us decide to use the much faster *simple* negation detection method for the remaining of our experiments.

B. Sentiment Lexicons Evolution over Different Years

We then decide to conduct an analysis on how sentiment lexicons and word sentiment polarity scores can change over the years. In order to do this we use as input corpus the Amazon Books dataset, which contains reviews that date all the way back to 1996 and span until 2018. We group reviews in 3 different subgroups, based on the year they were written in:

- the first group contains reviews from 1996 to 2004;

Years group	1995 - 2004		2009 - 2013		2016 - 2018	
Position	**token**	**score**	**token**	**score**	**token**	**score**
1	disservice	-1.000	purports	-1.000	downhill	-1.000
2	disappointing	-0.979	forgettable	-0.910	chore	-0.879
3	yawn	-0.976	downhill	-0.879	yawn	-0.858
4	unengaging	-0.965	underwhelmed	-0.871	disappointing	-0.849
5	underwhelming	-0.964	disappointing	-0.827	mediocre	-0.848

TABLE V: The top 5 negative scoring words per groups of years, after being scaled in the range $[-1, 1]$.

- the second group contains reviews from 2009 to 2013;
- the last group contains reviews from 2016 to 2018.

In order to save on computation time, we decide to upper bound the size of the input corpus at 2.5 million reviews, thus obtaining three groups of size respectively 700k, 2.5 million, 2.5 million. We use the same method explained in Section II to generate three lexicons, using as vocabulary the same one from GloVe CommonCrawl word vectors. The generation of the single lexicon, except for the input corpus, leverages the same parameters and settings used for the experiment described in Section III-A.

We report the number of positive and negative words that we obtain for each group of years in Table IV, considering words that have a score that lies between -0.2 and 0.2 as neutral. It is interesting to note that in the second group the amount of negative words almost doubles from the first and the third ones, which are between them of very similar magnitude.

In order to attest how much consistency there is between the years in terms of most polarizing words, we report the 5 most negative words for each group of years in Table V. As we can see, all of them, if used to describe how much one has enjoyed reading a book, convey a very negative feeling.

We finally report our most interesting findings in terms of words that have changed their score from positive to negative or vice versa during the different years. One very clear example of how domain specific sentiment lexicons can change over the years comes from the word *Grey*: in the two lexicons derived from the first two group of reviews, this word presents a very neutral sentiment polarity score (-0.03 in both of them), since it is mostly associated with the concept of the colour grey, which has no inherent clear sentiment polarity. But, because of the burst in popularity of the book *Fifty Shades of Grey* around 2012 that escalated even more with the publishing of the homonymous movie in 2015, we can clearly see a big polarity change of the word *Grey* in the third lexicon (0.11), meaning that this word started being used in mostly positive reviews that described probably the actual book or similar ones to it.

The same behaviour can be seen from the word *Sheldon*, that in the first two groups is very neutral (0.09, 0.05), but in the third group shows a big jump in polarity towards positivity (0.2), probably because of the big burst in popolarity of the show *The Big Bang Theory*, in which *Sheldon* Cooper is one of the main protagonists. It is interesting to note that this method picks up this behaviour solely from receiving as inputs reviews of books. Other words also like *Escobar* portray the same behaviour: from being mostly neutral in the first two groups (0.00 and 0.05) it shows a big jump in the third one (0.42), possibly because of the massive popularity that the Netflix show *Narcos*, which started airing in 2015, had in mass media culture during those years.

Reddit Community	Positive	Negative
r/Conservative	20169	1108
r/Feminism, r/AskFeminists	10421	1767

TABLE VI: The number of positive and negative words for each lexicon generated from comments and posts of the different communities, considering words with a score $S < -0.15$ or $S > 0.15$ as non-neutral.

These examples show that changes in word meaning coming from different domains (like the one of series tv in these two last cases) because of sudden raise of popularity in mass media culture can actually influence other domains that are not directly related to what made the word meaning shift in the original domain.

C. Reddit Communities Lexicons Comparison

Finally we try to apply the same methodology for lexicon generation described in Section II using as input corpus comments and posts from Reddit communities. We use Cornell's ConvoKit in order to obtain utterances from a specific subreddit and then we consider as positive comments and posts that receive a total difference between upvotes and downvotes higher than 1, and as negative the ones with total difference lower than 1. We consider comments and posts with exactly 1 as difference between upvotes and downvotes as neutral, since in most subreddits new posts and comments start from 1 upvote, and we discard them. One could argue that probably a comment that has a difference between upvotes and downvotes of 5 should not have the same impact as a comment with a difference between upvotes and downvotes of 10k on the scores of the words used in it, and could explore better ways to separate between positive and negative comments: having more computational resources could have permitted us to manage much bigger subreddits containing millions of comments and posts, and we could have then considered for example only reviews with more than 100 upvotes minus upvotes as positive and with less than -100 upvotes minus downvotes as negatives.

We also discard comments and posts which were deleted by mods or users themselves, that are present in the corpus as *[deleted]* or *[removed]*, since previous experiments where we kept them rightfully shifted most of the negative polarity scores to the words *deleted*, *removed* and their synonyms.

Instead of using the original GloVe CommonCrawl word embeddings, for this experiment we leverage GloVe word

token	Feminist	Conservative	token	Feminist	Conservative	token	Feminist	Conservative
Arma2	-0.17	0.00	*Aspartame*	-0.15	0.00	*BigBrother*	-0.16	-0.02
Colonize	-0.15	0.01	*Combustibles*	-0.17	-0.03	*Evangelism*	-0.15	-0.05
Fellate	-0.19	0.10	*HYDROCHLORIDE*	-0.17	0.08	*Insigne*	-0.16	0.00
Islamphobe	-0.15	0.03	*Lacazette*	-0.16	0.03	*melodramatic*	-0.16	0.04
Molestar	-0.16	0.12	*PLAYERUNKNOWN*	-0.17	0.03	*Reproduce*	-0.16	0.01
SATIRE	-0.15	0.07	*chigga*	-0.16	0.10	*childish*	-0.18	-0.07
crybaby	-0.16	-0.15	*flamebaiting*	-0.17	0.00	*shitehole*	-0.16	-0.04
commonsense	0.00	-0.17	*IslamQA*	-0.02	-0.16	*Killary*	-0.09	-0.15
LIBTARD	0.03	-0.33	*LIBTARDS*	-0.09	-0.26	*NEVERTRUMP*	0.06	-0.18
NeoLiberal	0.02	-0.15	*SUCKERS*	0.03	-0.17	*conservatards*	-0.07	-0.34
conspitard	-0.07	-0.17	*denialist*	-0.03	-0.15	*dork*	0.08	-0.16
fanbois	-0.03	-0.26	*gotem*	0.13	-0.16	*Hussein*	-0.03	-0.18
inbreds	-0.09	-0.15	*kiddo*	-0.05	-0.15	*republitards*	-0.06	-0.19

TABLE VII: Some interesting findings in terms of words with a negative score in a community and non-negative in the other. Note that we consider as non-neutral words with a score $S < -0.15$ or $S > 0.15$.

token	Feminist	Conservative	token	Feminist	Conservative	token	Feminist	Conservative
SOPHIE	0.16	0.04	*supportive*	0.18	0.02	*Threesome*	0.22	-0.03
Vegetarian	0.15	0.03	*Veganuary*	0.18	0.04	*Xenophilie*	0.17	0.02
Yaay	0.17	-0.06	*breastfeeding*	0.21	-0.05	*genderqueer*	0.16	0.05
gynecologist	0.21	-0.02	*masturbator*	0.15	-0.02	*miscarriage*	0.15	-0.01
nsfw	0.38	0.05	*oopsie*	0.22	-0.03	*polyamory*	0.15	0.04
polysexual	0.19	0.00	*postpartum*	0.21	-0.01	*psychologist*	0.18	0.01
psychotherapist	0.18	0.02	*transgirl*	0.15	-0.03	*AssassinsCreed*	0.03	0.17
Atalanta	-0.03	0.18	*Catfight*	0.01	0.17	*Creepiness*	-0.09	0.17
Cuomo	0.01	0.16	*Degradation*	-0.21	0.22	*Drogba*	-0.01	0.18
Elitism	-0.03	0.15	*Fempire*	-0.1	0.17	*Foxsports*	0.04	0.16
GamingCirclejerk	-0.04	0.16	*HunterxHunter*	0.03	0.16	*Hypermasculinity*	-0.04	0.15
mansplaining	0.03	0.20	*manspreading*	0.01	0.16	*Masculinity*	0.01	0.16
Patriarchy	0.00	0.20	*Squeaker*	-0.07	0.016	*TikTok*	-0.15	0.21

TABLE VIII: Some interesting findings in terms of words with a positive score in a community and non-positive in the other. Note that we consider as non-neutral words with a score $S < -0.15$ or $S > 0.15$.

vectors fine-tuned on a Reddit corpus of comments[5]. The rest of the parameters are the same as the previous experiments.

Similarly to [5], we want to create lexicons for different Reddit communities, but unlike them, our lexicons do not report the sentiment polarity score of words: instead, since we consider the difference between upvotes and downvotes of comments and post to separate them between positive or negative, our scores portray how much a word w_j impacts on the likelihood of the comment containing it of receiving a positive or negative number of upvotes minus downvotes in the considered community. Thus, a word w_j that definitely has a positive connotation in one subreddit but a negative one in another, could still end up with a similar score because in the one where it is seen as negative, it is mostly used in comments that portray it in a negative way (thus receiving a lot of likes from people that share the same opinion), while in the one where it is seen as positive, it is mostly used in comments that portray it in a positive way. Still, it is fairly interesting to see what happens when lexicons for two opposite communities are built using our method and what words happen to land with different scores between two contrasting communities.

We decide to concentrate our study on two different communities:

- the first one comprise of two different subreddits (**r/Feminism** and **r/AskFeminists**), of which we merge their posts and comments, removing the neutral and the deleted ones. At the end of this process, we end up with 602k posts plus comments.
- the second one is **r/Conservative**. We again remove neutral and deleted comments, and end up with 2.13 million posts plus comments.

[5]https://github.com/leigh-plt/glove.reddits

After creating the two lexicons using the method reported above, we obtain the number of positive and negative words reported in Table VI. Even if the mean of both lexicons is 0.0, we can notice that the *r/Conservative* lexicon presents almost double the positive words of the other community, and almost 33% less negative words. This could very well be due to the fact that *r/Conservative* is more of a debate subreddit, where many different topics get discussed without getting downvoted from the start, while *r/Feminism* and *r/AskFeminists* seem to prefer talking about topics concerning womanhood and the feminist movement, disregarding many other different topics.

Also, the quite higher number of positive words with respect to the negative ones could be due to the fact that most communities act like a bubble, in which it is harder to find comments that go against the main sentiment of the community itself about a given topic. Many subreddits in fact act more as a community where users meet people with similar ideas to them, instead of being a place to debate topics.

We report some interesting findings in terms of words that end up with a negative score in one community and non-negative in the other one in Table VIII, and ones that end up with a positive score in one community and non-positive in the other one in Table VII.

From both tables we can assume the motive behind the different scores of some tokens from the clear opposite views that these two communities have about them: for example, words like *republitards* and *conservatards* are very negative in *r/Conservative*, since they are probably mainly used as insults from non-members of the community. Others like *Hypermasculinity* and *Patriarchy* instead have a fairly high positive score, due to upvotes to comments that talk positively about these topics, or also use them sarcastically in order to mock opposite views about them.

We also find some interesting patterns while scraping the results: we notice for example that many footballers surnames like *Insigne* and *Lacazette* end up with a negative score for the *r/Feminism* and *r/AskFeminists* communities, probably due to users of these communities not being interested in football related topics and personalities. Other results are bizarre and quite harder to explain: we notice a vast presence of *Pokémon* names, like *Chikorita* and *Vulpix*, being labeled as positive for the *r/Conservative* community, while being neutral for the other one. This may happen because generally *r/Conservative* users are more interested in gaming related topics and may upvote gaming related comments. In fact, we can also notice for example the word *AssassinsCreed* having a high score for this community.

Another interesting pattern we can notice is that many different variations of the laughing word *hehehe* or of the positive exclamation *yaay* end up as negative or neutral in *r/Conservative* while positive in *r/Feminism* and *r/AskFeminists*, probably due to a different style of writing between users of the two subreddits.

Finally, many words that are more related to concepts dear to the feminist world like sexual empowerment or mental health acceptance end up with a quite high score in its lexicon: from Table VIII, we can see that words like *nsfw* (not safe for work, usually related to erotic posts), *psychotherapist*, *polysexual*, *transgirl*, *genderqueer*, *miscarriage* have a positive score in the first lexicon but neutral in the second one.

With these community specific generated lexicons, one can then predict if a new comment or post is going to be positively or negatively received in the relative community.

IV. Conclusions

In conclusion, as we saw from Section III-A domain specific sentiment lexicons work quite well in performing Sentiment Analysis on unseen text that refers to the same domain. We must recognize though that they probably work best for domains for which the context of the text being classified is not crucial to the sentiment that the phrase conveys. Instead, for more context-dependent domains, such as the one of Sarcasm Detection, a lexicon generated with examples labelled as sarcastic or not would not give us the same results.

It would be interesting then to expand on the sentiment lexicon concept by considering the word scores not as fixed values but rather as random variables that can vary based on the context in which they appear.

References

[1] SM Mazharul Islam, Xin Dong, Gerard de Melo, Domain-Specific Sentiment Lexicons Induced from Labeled Documents, Proceedings of the 28th International Conference on Computational Linguistics, 2020, pp. 6576–6587.

[2] Sascha Rothe, Sebastian Ebert, Hinrich Schutze, Ultradense word embeddings by orthogonal transformation, Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016.

[3] Pennington Jeffrey, Socher Richard, Manning Christopher, GloVe: Global Vectors for Word Representation, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.

[4] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts, Learning word vectors for sentiment analysis, Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 2011, pp 142—150.

[5] William L. Hamilton, Kevin Clark, Jure Leskovec, Dan Jurafsky, Inducing Domain-Specific Sentiment Lexicons from Unlabeled Corpora, CoRR, 2016.