university of groningen / faculty of arts

Domain adaptation in sentiment analysis

combining cross-domain data for better text classification

Steven Kema

Bachelor thesis
Informatiekunde
Steven Kema
2068222
June 13, 2016

ABSTRACT

In this paper we investigate the usefulness of combining data from different domains as a way to accomplish domain adaptation in sentiment analysis. Text classification is carried out through the Naive Bayes algorithm and is implemented through Python Nltk. The data used consists of user-generated game reviews web-scraped from online gaming platform Steam. Genre categories of these reviews are used as different domains. Having a training set consist of 70-90% of genre Sports and rest other genre yields the best results for an overall good classification system, with an average accuracy score around 0.75. A potential reason for this observation is that the feature set for Sports is relatively sparse and a result lends itself as a good base set for sentiment analysis. Additionally, domain-dependent behavior is observed in using genres for different domains, which might mean that a smaller differentiating factor such as genre can be utilized in overcoming the domain adaptation problem.

CONTENTS

1 INTRODUCTION

How do people on Twitter feel about the current political climate? Can we predict how financial markets will move based on news reports and social media sentiment? Is it possible to automatically get insight into customer satisfaction through user reviews? These are all questions that have to do with a technique called sentiment analysis. Sentiment analysis, also known as opinion mining, is a rapidly developing field of study that analyzes people's opinions, sentiments, and emotions towards a whole range topics (Liu, 2012).

Sentiment analysis has since the early 2000s grown to be one of the most active research areas in natural language processing, yet is still a relatively young field with a long way to go, and progress to be made (Liu, 2012). This small academic base is predominantly due to a lack in availability of opinionated text before the rise of the World Wide Web (Liu, 2010). Platforms such as Twitter, discussion forums, and blogs nowadays provide a vast amount of user-generated content that can be collected, processed and used for scientific research (in this case sentiment analysis/opinion mining).

According to Liu (2012) it has been shown that sentiment classification is highly sensitive to the domain from which the training data is extracted. The term domain is used to delimit a certain category of opinion documents; example categories being movie reviews, fincancial reports, legal text or scientific articles. Training a classifier on opinionated text from one defined domain will therefore perform relatively bad when testing it on text coming from a different domain, compared to testing on its own training domain. This is because words and used in the different domains for expressing opinions can be used differently. For example a word can have a positive meaning in one domain, yet may be negative in another (Loughran and McDonald, 2011). Being able to build a text classifier that works well on multiple types of text is therefore very useful, because it can save a lot of time and effort. This is due to the fact that collecting data, processing, and most importantly annotating the test, development, and training dataset accurately is quite labor intensive. Research on domain adaptation tries to solve this inability of using a broader domain for classification. Domain adaptation (also called domain transfer) refers to the problem of adapting a statistical classifier trained on data from one (or more) source domains to a different target domain (Prettenhofer and Stein, 2010).

In this paper the effectiveness of combining data from different domains as a domain adaptation technique will be examined using user-generated game reviews as data. The guiding research questions will be as follows:

1. Can we observe domain dependency in sentiment analysis between user reviews of different game genres?
2. Does the process of combining in- and out-domain training data result in an overall better sentiment classification system?
3. Is Steam user reviews a useful data source for research in transfer learning and sentiment analysis?

The data used for the analyses will come from the online gaming platform Steam [1]. Steam gives users the ability to purchase and download video games. After purchasing a game these users are given the option of reviewing their bought games, in which they have to give a sentiment about the game (either positive or negative). This gives us a new large, annotated, and accessible data set that can be used to determine user sentiment.

The accepted contemporary application of sentiment analysis is through the use of Machine Learning algorithms. These algorithms and be divided into two/three categories: supervised, semi-supervised, and unsupervised. A classifier is trained on a training set, then tested on a separate testing set of opinions. During testing the algorithm will try to predict whether a text is positive or negative based on data presented in the training set. In this paper Naive Bayes will used to classify reviews as either positive or negative.

The domain approach of this paper is to divide user reviews into smaller groups (based on game genre)to see if they can be used in the field of domain transfer/adaptation. These smaller groups have the advantage of having decent enough overlap with each other in terms of domain related words. These categories will act as sub-domains, and should help us in training a classifier that makes use of domain adaptation techniques. If this approach proves to be successful in improving the detection of sentiment in cross-domain text classification, it will provide researchers a useful new dataset to train and test domain adaptation techniques on. The main goal of this paper however is to observe domain dependent behavior in sentiment analysis, and to see whether combining different domains improves sentiment classification systems.

[1] http://store.steampowered.com

2 BACKGROUND

In literature different approaches to sentiment analysis and domain adaptation are explored. Pang et al. (2002) were the first ones to take the approach to classify movie reviews into two classes, positive and negative. It was shown that using unigrams (a bag of words) as features in classification performed quite well with either Naive Bayes or SVM (Liu, 2012). An especially interesting application can be found in using the genre of games and their reviews in combination with domain adaptation. In sentiment analysis, classification systems are usually built for a specific domain, such as movie reviews (Pang et al., 2002), congressional floor debates (Thomas et al., 2006) or product recommendations (Snyder and Barzilay, 2007).

One such a way is through Structured Correspondence Learning (SCL), proposed by Blitzer et al. (2007). In the experiment they make use of a corpus of reviews for four different types of products from online retailer Amazon: books, electronics, DVDs, and kitchen appliances. SCL's working is based on the choosing of certain pivot features. These are features that are used to link source and target domain.

Aue and Gamon (2005) on the other hand choose a different approach to tackling the adaptation problem. For the domain adaptation problem four approaches are presented by Aue and Gamon (2005). The first one is to train on a mixture of labeled data from other domains. The second approach is to limit the set of features to those observed in the target domain. The third way is to use ensembles of classifiers from domains with available labeled data. The last approach is to combine small amounts of labeled data with large amounts of unlabeled data in the target domain. In their research they collect their data from four different sources: movie review data, book review data, Product Support Services web survey data, and Knowledge Base web survey data. (Aue and Gamon, 2005) have chosen these four sources based on the property differences between text types. They note that movie reviews tend to be lengthy and elaborate; book reviews are shorter but still may consist of multiple paragraphs, and that the two sets of survey data, on the other hand, consist of typically very short pieces of text. In contrast this experiment will make use of data that has a relatively uniform structure between domains, and focuses mostly on linguistic, and context-dependent differences pertaining to sentiment analysis. This is possible due to sourcing the data from only one platform, and distinguishing domains based on the different genres.

3 DATA AND MATERIAL

3.1 COLLECTION

The data that is needed to perform the experiment consists of user-generated reviews from online platform Steam. The collection of review data is done by web scraping the reviews off the Steam Store website. This approach is chosen due to the fact that even though Steam has an API, it unfortunately does not provide an option to obtain user reviews in this way. Using a web scraper is therefore the next best thing. Each game sold on the platform has its own web page; each store page showing a selection of user reviews for the particular game.

The way web scraping works is that an html page (could also be xml) is requested and parsed for specified content. Relevant content is then returned. A straightforward approach would be to parse the page from the following link for each game:

```
http://store.steampowered.com/appid/[id-code]
```

The `[id-code]` is the unique identifier for each game. Unfortunately due to the way reviews are provided by the website it is not possible to scrape in this manner. The workaround is to access the reviews via an alternative url, namely:

```
http://store.steampowered.com/appreviews/{0}?start_offset={1}&day_range
    ={2}&filter={3}&language={4}
```

The review scraper is written in Python and uses Requests combined BeautifulSoup to collect the user reviews. Requests is an HTTP library for Python that allows you to connect to web pages in order to pull data from it. The code snippet below shows how Requests is used in the scraper.

```
import requests
url = 'http://store.steampowered.com/appreviews/{0}?start_offset={1}&
    day_range={2}&filter={3}&language={4}'
url = url.format(app_id,offset,day_range,filter_type,language)
r = requests.get(url)
json = r.json()
```

The script loops over this part of the code, requesting and scraping the url page, then increasing the offset by 25 and repeating the process. Increasing the specified offset variable in the url allows the script to scrape a set of new reviews, either until a specified maximum amount of reviews is reached, or until there are no more reviews left for the game.

BeautifulSoup is a library for pulling data out of HTML and XML pages. It finds all review blocks `<div class="review_box">..reviews..</div>` in the provided json content and looks for each review's user given sentiment and review text. Additional data such as username and date of writing can be collected in exactly the same way, but since this data wasn't used in the experiment it was omitted to increase speed of the scraper.

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(json["html"], "lxml")
review_box = soup.find_all("div",class_="review_box")
```

```
4 for review in review_box:
5     review_text = review.find("div",class_="content").get_text()
6     sentiment = review.find("div",class_="title ellipsis").get_text()
```

Scraped content is returned and is written to a .csv file with the use of the csv.writer() module. All reviews for a particular game are written to one .csv file and are categorized based on the genre assigned to them by Steam. The chosen genres are: action, simulation, sports, strategy. The dataset contains around 320000 user reviews in total. The distribution is not equal however, due to the fact that some genre domains have less reviews over the whole genre. To correct this, the amount of reviews used in the analysis is based on the genre with the lowest review count.

3.2 ANNOTATION

In order to perform text classification through through supervised learning, an annotated dataset will be needed for the training of a classifier. This dataset will consist out of game review content and the accompanying sentiment of each review. This sentiment can be either 'Recommended' or 'Not Recommended', and is provided by the users that write and publish their review. A user is only allowed to publish a review when they include a sentiment, so all reviews will contain a specified sentiment by default. An example of a Steam user review can be found below (figure 1). Category distinction is based on genre, and is selected by hand. Each game belongs to one or more genres: a primary genre is selected. Since annotation is already part of the data collection it can be considered automatically generated. Its annotation quality is therefore not Gold Standard (GSC), but falls under Silver Standard Corpora (SSC) (Wissler et al., 2014).

Figure 1: An example of a positive user review on Steam.

3.3 PROCESSING

Since users can only review the games that they've bought not many of the reviews will be spam. Also since each reviewer must give their review a sentiment, which is either positive or negative, annotation is already provided and no processing for this is needed. Connecting the user sentiment with their review will probably give an accurate enough classification. One important thing is that the reviews should all be written in the same language;

I have chosen English since this will give the largest dataset. Since the Steam platform already filters their reviews based on language, the expectation is that most collected reviews will be in English already. In order to filter out any non- english reviews the language option built into nltk's word tokenizer is used, which uses both the TreebankWordTokenizer (based on Penn Treebank) and PunktSentenceTokenizer. The code snippet below shows the usage of the module.

```
from nltk.tokenize import word_tokenize
tokenized_review = word_tokenize(review, language='english')
```

Loading data into the classifier script can be seen in the snippet below. Reviews are taken from the directory path's .csv files and put into either a negative or a positive list, depending on the given user sentiment.

```
def load_dataset(path, genre, limit):
    file_directory = join(path, genre)
    print("Loading reviews from: {}".format(file_directory))
    reviews_positive = []
    reviews_negative = []
    for file in glob.glob(os.path.join(file_directory, '*.csv')):
        with open(file, "r", encoding="utf-8") as f:
            reader = csv.reader(f)
            #skip the first line as it only contains header information
            next(reader, None)
            for line in reader:
                if line[6] == "Recommended":
                    reviews_positive.append((get_features(line[5]), "p"
                        ))#p:positive
                else:
                    reviews_negative.append((get_features(line[5]), "n"
                        ))#n:negative
                if len(reviews_positive) >= limit and len(
                    reviews_negative) >= limit:
                    return review_positive, reviews_negative
    return reviews_positive, reviews_negative
```

4 METHOD

4.1 TOOLS

Since the dataset is fully labeled, supervised learning will be the most logical approach for this experiment. The used sentiment classification algorithm will be Naive Bayes, which is a Bag of Words method. Since this experiment's focus is on observing domain dependency together with domain adaptation, and not on improving a classifier, the type classification algorithm isn't further explored. It is therefore kept the same across the whole experiment in order to achieve constant and valid results.

The machine learning and natural language aspects will be implemented with the help of Python Natural Language Toolkit (NLTK). It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, and parsing (NLTKProject, 2016). The code example shows the implementation of Nltk's Naive Bayes classifier.

```
1 from nltk.classify import NaiveBayesClassifier
2 #train the classifier
3 classifier_a = NaiveBayesClassifier.train(labeled_a)
```

Evaluation will be done with the use of accuracy scores. Accuracy ranges from 0 to 1 and is based on the classifier's amount of correctly predicted reviews compared to the overall amount. A better performing system will as a result produce a higher accuracy score than a worse system. The formula below shows how accuracy is calculated.

$$\text{Accuracy} = \frac{\text{true positive} + \text{true negative}}{\text{true positive} + \text{false positive} + \text{false negative} + \text{true negative}}$$

True positive and true negative are the amount of correctly predicted reviews: true positive for "Recommended" and true negative for "Not Recommended". False positive and false negative are the amount of reviews of which the sentiment was predicted incorrectly.

4.2 APPROACH

4.2.1 Part 1

Using n-grams instead of single tokenized words in a bag of words method can help increase the accuracy score of a classifier. The classifier script will load two different genre datasets, train two classifiers, then testing both in- and out-domain. The process is then repeated for a different n-gram. This will be done for an n-gram range between 1 to 5. Afterwards the out-domain accuracy scores will be combined into an average accuracy for each classifier, making it possible to see which n-gram performs best overall. Also an in-domain graph will be plotted. In order to find the optimal n-gram, these steps will be repeated for different genre combinations. The amount of

reviews for each genre will be the same, and kept constant for the other experiment parts. Each genre will have a dataset of 8000 reviews, half of them positive, and the other half negative. For testing 10% of the reviews will be taken from the dataset before using the rest of them for training.

4.2.2 Part 2

The goal of this part of the experiment is to find out if we can observe domain-dependent behavior between two different genres. After deciding on the overall best n-gram from part 1, accuracy score analysis of out-domain classification will take place for this best n-gram. Accuracy scores from part 1 of the method will be used for this.

4.2.3 Part 3

In order to find out if combining data from two different domains results in an overall superior classifier, the following approach will be taken. The first step is to see how the two domains perform separately, on their own as well as the other domain. This is to get an understanding of each domain's baseline performance. The next step is to mix the training set to be used on the classifier. The choice has been made to mix data from the two domains with a linear ratio, starting from 10% to 90% using steps of 10%. This will give us a relatively detailed overview on the effects of combining training data . Accuracy scores will be recorded to see if a trend can be observed from the combining of data originating from two different domains. This experiment will be performed for all combinations of the four genres.

5 RESULTS AND DISCUSSION

5.1 CHOOSING THE BEST N

The classification accuracies resulting from in-domain testing and training are shown in table 1, and are visualized by genre in figure 2. The in-domain accuracy scores of the strategy domain are the highest for all n-grams. The simulation genre performs the worst overall, even though its score is a bit higher than Sports at N = 3. More generally the classifiers seem to benefit the usage of both bigrams and trigrams, compared to their unigram baselines. At N = 4 however, performance goes down. Another thing that can be observed from the data is that in general a higher N makes the classifiers for the different genres perform more similar. At N = 5 the accuracy scores for each genre are almost all the same. A reason for this could be that 5-grams are too sparse and don't provide enough information for the classifier to be able to predict properly. This seems to even out to an accuracy score around 0.61. Trigrams perform arguably the best of all five, due to the fact that accuracy scores stay quite high, yet also achieve relatively similar scores for each genre. Bigrams could prove to be a relevant alternative.

Table 1: Accuracy scores of in-domain sentiment analysis

Genre	Ngram				
	1	2	3	4	5
Action	0.72375	0.76625	0.79625	0.6925	0.615
Simulation	0.65875	0.7025	0.75625	0.64125	0.6125
Strategy	0.81875	0.8575	0.81625	0.74875	0.63
Sports	0.69	0.79875	0.7175	0.69	0.6187

In table 2 and figure 3 below show the averaged out-domain accuracy scores for each training domain, e.g. classifier is trained on action then tested on simulation, strategy and sports. The three resulting accuracy scores are then averaged to see which classifier performs best for out-domain testing, combined with the selected n-gram. The graph shows that strategy and sport perform comparable, and give higher average accuracy scores than action and simulation. This could be an indication that these two provide an overall better base for cross-domain classification (relevant for part 2 and 3 of the experiment). The best n-grams are again for N=2 and N=3. Between these two, action's score is a bit higher for trigrams, but overall there's barely any difference between the two. Similarly scores go down again for N greater than 3. In the further parts n-grams will be set at N=3

Figure 2: in-domain n-gram classification

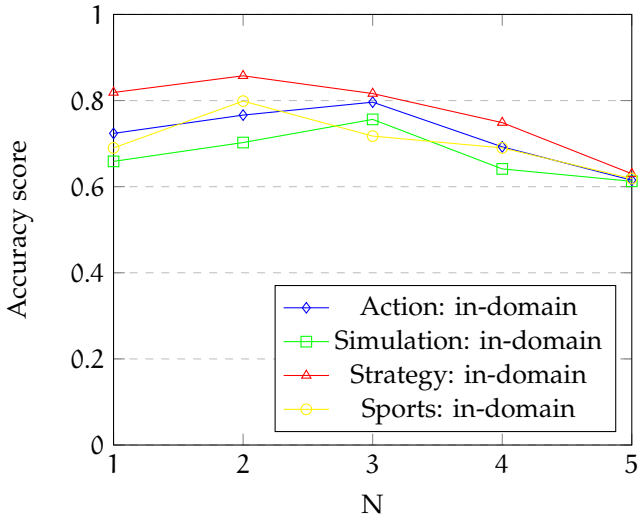
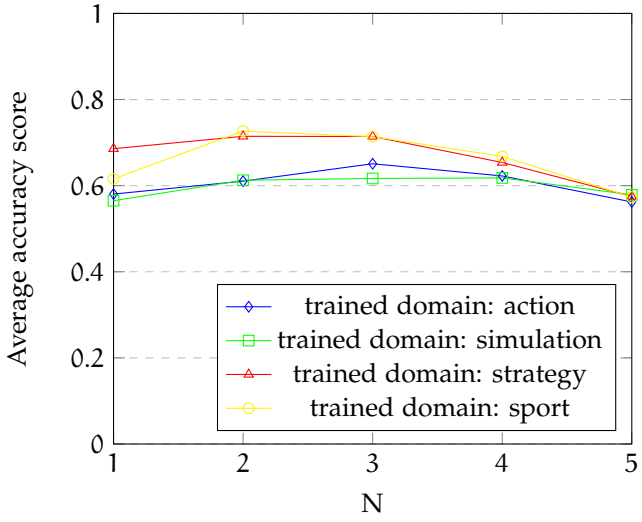


Table 2: Out-domain averaged n-gram accuracy scores

Genre	Ngram				
	1	2	3	4	5
Action	0.580416667	0.610416667	0.650833333	0.6225	0.562083
Simulation	0.565	0.6125	0.616666667	0.617916667	0.57875
Strategy	0.685833	0.714583333	0.714166667	0.65375	0.575
Sports	0.615833333	0.726666667	0.71375	0.667916667	0.573333333

Figure 3: Out-domain averaged n-gram accuracy scores



5.2 OBSERVING DOMAIN-DEPENDENT BEHAVIOR

Below in table 3 the accuracy scores are set out against each other for trigrams. The first thing that can be noticed is that in-domain training and testing results in the highest accuracy scores for action, simulation and strategy. The exception to this is Sports, which is somewhat unusual because normally you'd expect that in-domain training and testing works the best. A classifier trained on Sports games reviews gives the best result when tested on Strategy, giving an accuracy score of 0.75375. The reason for this could be that reviews on Sports games provide a more general set of features, giving us an overall more sparse classifier. All out-domain testing sets have the lowest accuracy scores when the classifier is trained on Sports.

Table 3: Complete table of accuracy scores, N = 3

		Testing			
		Action	Simulation	Strategy	Sports
Training	Action	0.79875	0.6725	0.6375	0.6425
	Simulation	0.65625	0.7675	0.61375	0.58
	Strategy	0.68625	0.71875	0.81625	0.7275
	Sports	0.68	0.7175	0.75375	0.7175

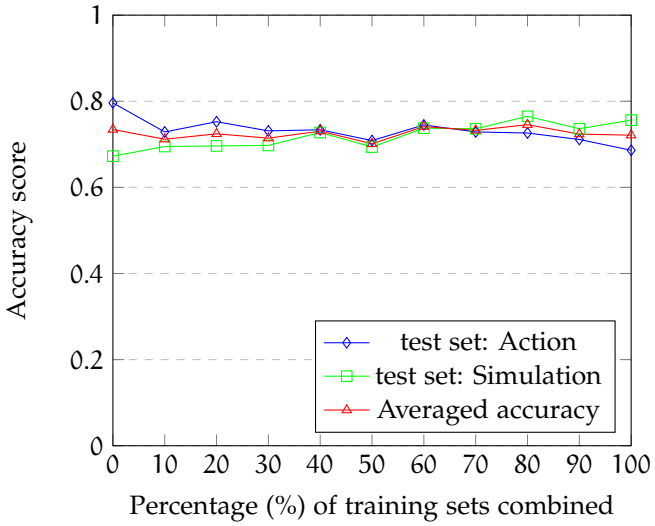
5.3 COMBINING DATASETS

In the figure below (figure 4) the accuracy scores are plotted for test sets from genre Action and Simulation, and an average between the two. On the x-axis the percentages of combined training sets are presented. At a percentage of 0, the training set is unmixed, and contains only reviews from the Action domain. At 100% the training set only contains reviews from Simulation. At these points the accuracy scores for their respective test sets are the same as their usual in-domain scores. It is therefore no surprise that the Action test set performs so well at 0% train mixed, and the other way around for Simulation. At a 50/50 training set mix, both test sets perform quite similar. Interestingly enough this also goes for 40/60 or 60/40, but with the added effect that they both give better accuracy scores than at 50%.

Table 4: Combined datasets: Action- Simulation

Genre	Percentage combined										
	0	10	20	30	40	50	60	70	80	90	100
Action	0.79625	0.72875	0.7525	0.73125	0.73375	0.70875	0.745	0.72875	0.72625	0.71125	0.68625
Simulation	0.6725	0.695	0.69625	0.6975	0.7275	0.69375	0.7375	0.73625	0.765	0.73625	0.75625
Average	0.734375	0.711875	0.724375	0.714375	0.730625	0.70125	0.74125	0.7325	0.745625	0.72375	0.72125

Figure 4: Combined datasets: Action- Simulation



A similar trend can be observed from the training set combination of Action - Strategy, with again an apparent sweet spot at the 40/60 and 60/40 ratio.

Figure 5: Combined datasets: Action- Strategy

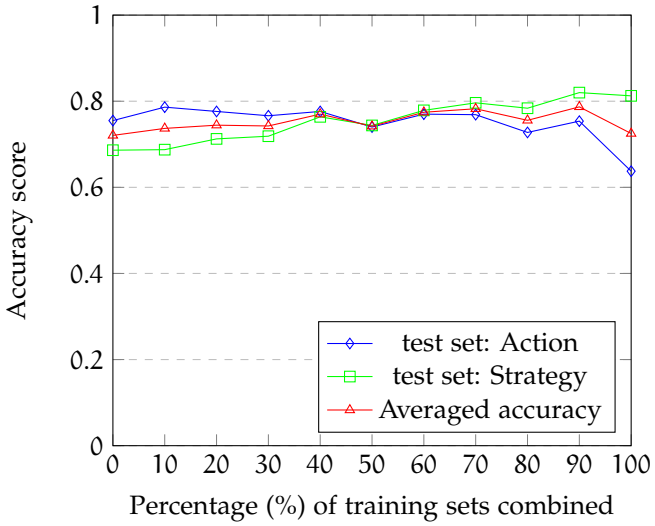


Table 5: Combined datasets: Action- Strategy

Genre	Percentage combined										
	0	10	20	30	40	50	60	70	80	90	100
Action	0.755	0.78625	0.77625	0.76625	0.77625	0.74	0.77	0.76875	0.7275	0.75375	0.6375
Strategy	0.68625	0.6875	0.7125	0.71875	0.76375	0.74375	0.77875	0.79625	0.78375	0.82	0.8125
Average	0.720625	0.736875	0.744375	0.7425	0.77	0.741875	0.774375	0.7825	0.755625	0.786875	0.725

Combining Action with Sports gives us a completely different plot (figure 6) compared to Action- Simulation and Action - Strategy. The best overall classifier seems to be at 70% Sports/30% Action, and with no convergence between 40-60%. If anything, looks like an inverted version of the

optimal percentages from the plots in figure 4 and 5, and shifted to the right.

Figure 6: Combined datasets: Action- Sports

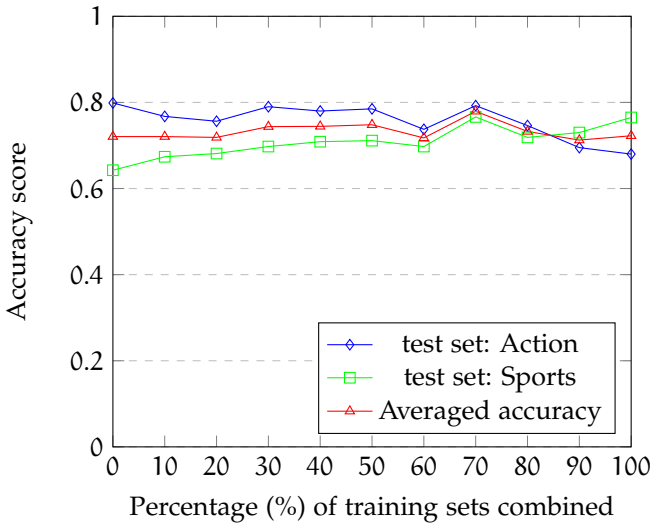


Table 6: Combined datasets: Action-Sports

Genre	Percentage combined										
	0	10	20	30	40	50	60	70	80	90	100
Action	0.79875	0.7675	0.75625	0.79	0.78	0.785	0.7375	0.7925	0.74625	0.695	0.68
Sports	0.6425	0.67375	0.68125	0.6975	0.70875	0.71125	0.6975	0.76625	0.71875	0.73	0.765
Average	0.720625	0.720625	0.71875	0.74375	0.744375	0.748125	0.7175	0.779375	0.7325	0.7125	0.7225

The plot for Simulation - Sports (figure 7) looks quite similar to figure 6 in its accuracy progression and combination percentage trend. The only difference here is that optimal combination percentage is shifted to 80% Sports, compared to 70% Sports in figure 6.

Figure 7: Combined datasets: Simulation- Sports

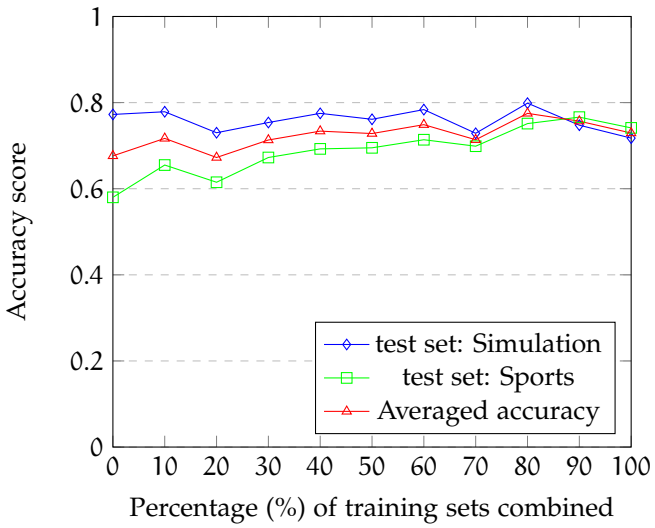


Table 7: Combined datasets: Simulation- Sports

Genre	Percentage combined										
	0	10	20	30	40	50	60	70	80	90	100
Simulation	0.7725	0.77875	0.73	0.75375	0.775	0.76125	0.78375	0.72875	0.79875	0.7475	0.7175
Sports	0.58	0.655	0.615	0.6725	0.6925	0.695	0.71375	0.69875	0.75125	0.76625	0.74125
Average	0.67625	0.716875	0.6725	0.713125	0.73375	0.728125	0.74875	0.71375	0.775	0.756875	0.729375

Figure 8 shows no distinct best percentage like the other plots. Only at 30% the accuracy scores for both test sets are very similar, however the average between the two is not the highest at this percentage. The highest average accuracy is at 90%, with a score of 0.7775

Figure 8: Combined datasets: Simulation- Strategy

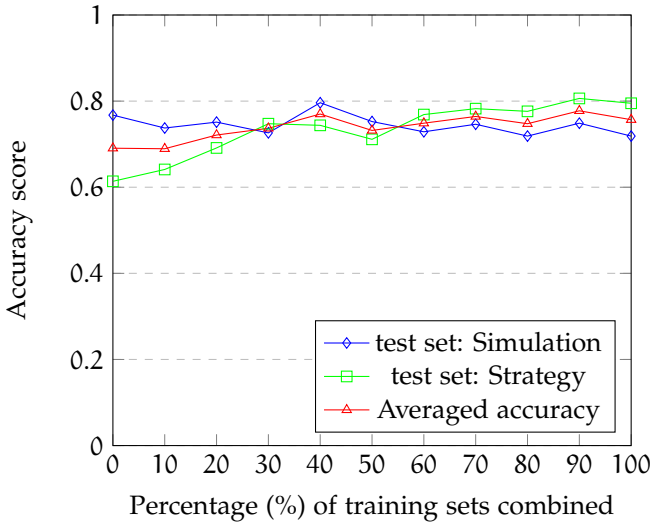


Table 8: Combined datasets: Simulation- Strategy

Genre	Percentage combined										
	0	10	20	30	40	50	60	70	80	90	100
Simulation	0.7675	0.7375	0.75125	0.72625	0.79625	0.7525	0.72875	0.74625	0.71875	0.74875	0.71875
Strategy	0.61375	0.64125	0.69125	0.7475	0.74375	0.71125	0.76875	0.7825	0.77625	0.80625	0.795
Average	0.690625	0.689375	0.72125	0.736875	0.77	0.731875	0.74875	0.764375	0.7475	0.7775	0.756875

The result of combining the genre Sports with Strategy can be seen in figure 9. Interestingly enough the Sports test set performs worse than the Strategy set at every percentage, even when training set is 100% Sports (which is in-domain). At 90% Sports and 10% Strategy the Sports set does best, having an accuracy score of 0.77. This is not the combination percentage with the highest overall accuracy score, however the difference between Strategy & Sports scores is minimal. Combining the two genres in this case only seems to benefit the classification of Sports reviews.

Figure 9: Combined datasets: Sports - Strategy

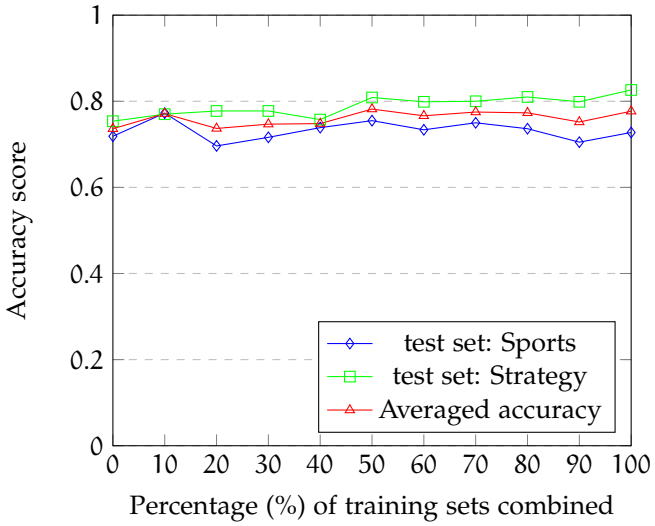


Table 9: Combined datasets: Sports - Strategy

Genre	Percentage combined										
	0	10	20	30	40	50	60	70	80	90	100
Sports	0.71875	0.7725	0.69625	0.71625	0.73875	0.755	0.73375	0.75	0.73625	0.705	0.7275
Strategy	0.75375	0.77	0.7775	0.7775	0.7575	0.80875	0.79875	0.8	0.81	0.79875	0.82625
Average	0.73625	0.77125	0.736875	0.746875	0.748125	0.781875	0.76625	0.775	0.773125	0.751875	0.776875

Ultimately there is no overall best combination percentage that can be observed from this experiment. One interesting part however is that every training set using the genre Sports seems to classify in a similar fashion. All combinations with a high percentage Sports and low percentage other genre seem to result in either a highest accuracy, or a good overall classifier with minimal score difference between the two test sets. A reason for this could be that reviews from the Sports genre maybe contain more general words comparatively, and as a result combined with a smaller amount of more specific features from another domain produce a relatively robust overall classifier. On its own, genre Sports produces a bad classifier, which can also be concluded from part 2 of the experiment. Used as a base classifier with more specialized features from another domain however it becomes quite good. Another observation is that a 40%/60% ratio (and vice-versa) for Action- Simulation and Action- Strategy provides a good overall classifier, yet gives worse results at a 50/50 combination. Maybe the feature set becomes too weak at this point which negatively impacts both genres.

6 CONCLUSION

Domain dependency can be observed in sentiment analysis between classifiers based on different game genres. It seems that user-generated reviews separated by their video game genres are distinct enough from each other to make a difference when it comes to sentiment classification. In the experiment nearly all genres gave the highest accuracy scores when in-domain classification took place. The only exception is genre Sports, which performs the worst both in- and out-domain, which might be due to the feature set being relatively sparse. From this can be concluded that treating video game genres as separate domains is a viable option when it comes to sentiment analysis in combination with domain adaptation/transfer learning research.

Combining data from different domains can result in an overall better sentiment classification system. Even though there seems to be no universal combination percentage for a best result, it seems that using a domain that is relatively sparse in features combined with a more specific domain does produce an improved overall system. This approach could prove useful for future research into domain adaptation.

Steam provides a useful dataset for sentiment analysis research, due to: volume, mandatory user annotated reviews, different genres to, and relatively good language categorization. For future work additional metrics and data are available, such as user profile information, user game information, helpfulness rated by others, review funniness (maybe use-case of detecting humorous language). A main limitation to Steam as a data source of user-generated opinionated text is that data is silver quality (labels are technically deduced from data, and not manually annotated). This could be fixed by manually checking the generated labels. Another issue is that most popular games are bought and reviewed more. If a game is deemed bad/unpopular no people will buy it, so no people review it. This makes it more difficult to get enough negative reviews. Overall Steam provides a relatively good quality, accessible dataset with lots of options for future research in sentiment analysis.

In the future sentiment analysis will in all likelihood evolve from relatively low complexity (positive/negative/neutral) to a deeper understanding and better predicting ability of sentiment and emotion. Being able to construct a good classifier that can predict user sentiment more broadly (broader domain or even multiple domains) will play an important role in developing such a system. New insights and improvements on domain adaptation will contribute to and facilitate these developments.

BIBLIOGRAPHY

Aue, A. and M. Gamon (2005). Customizing sentiment classifiers to new domains: A case study. In *Proceedings of recent advances in natural language processing (RANLP)*, Volume 1, pp. 2–1.

Blitzer, J., M. Dredze, F. Pereira, et al. (2007). Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *ACL*, Volume 7, pp. 440–447.

Liu, B. (2010). Sentiment analysis and subjectivity. *Handbook of natural language processing 2*, 627–666.

Liu, B. (2012). Sentiment analysis and opinion mining. *Morgan & Claypool Publishers*.

Loughran, T. and B. McDonald (2011). When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of Finance 66*(1), 35–65.

NLTKProject (2016). Nltk 3.0 documentation. Accessed: 2016-05-29.

Pang, B., L. Lee, and S. Vaithyanathan (2002). Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, Stroudsburg, PA, USA, pp. 79–86. Association for Computational Linguistics.

Prettenhofer, P. and B. Stein (2010). Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, Stroudsburg, PA, USA, pp. 1118–1127. Association for Computational Linguistics.

Snyder, B. and R. Barzilay (2007). Multiple aspect ranking using the good grief algorithm. In *HLT-NAACL*, pp. 300–307.

Thomas, M., B. Pang, and L. Lee (2006). Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, Stroudsburg, PA, USA, pp. 327–335. Association for Computational Linguistics.

Wissler, L., M. Almashraee, D. M. Díaz, and A. Paschke (2014). The gold standard in corpus annotation. In *IEEE GSC*.