Comparative Argument Mining

Mirco Franzek

June 26, 2018

Introduction

Comparative Argument Mining: An example

Given a sentence and two comparable objects like:

"Toyota is better than BMW at... providing reliable, economical auto transport."

we want to know if

- the sentence compares the first object and the second object
- the first object wins the comparison or
- the second object wins the comparison

Comparative Argument Mining: An example

Given a sentence and two comparable objects like:

"Toyota is better than BMW at... providing reliable, economical auto transport."

we want to know if

- the sentence compares the first object and the second object
- the first object wins the comparison or
- the second object wins the comparison

Comparative Argument Mining: An example

Given a sentence and two comparable objects like:

"Toyota is better than BMW at... providing reliable, economical auto transport."

we want to know if

- the sentence compares the first object and the second object
- the first object wins the comparison or
- the second object wins the comparison

Comparative Argument Mining: An example

Given a sentence and two comparable objects like:

"Toyota is better than BMW at... providing reliable, economical auto transport."

we want to know if

- the sentence compares the first object and the second object
- the first object wins the comparison or
- the second object wins the comparison

Related Work

- Little work on comparative argument mining
- Specific to a narrow domain, e.g. biomedical
- See [Fiszman et al., 2007], [Park and Blake, 2012] and [Gupta et al., 2017]
- Patterns and rule based systems

Creating a data set

Data Source

Needed Data

English pieces of text

1. with a high chance of being comparative
2. containing at least two known, comparable objects
   (Not like: "This is better than BMW ...")
3. understandable to many people

Objects, which are

1. comparable on at least on property
2. known by many people

Everything should be as domain unspecific as possible.

Needed Data

English pieces of text

1. with a high chance of being comparative
2. containing at least two known, comparable objects
   (Not like: "This is better than BMW ...")
3. understandable to many people

Objects, which are

1. comparable on at least on property
2. known by many people

Everything should be as domain unspecific as possible.

Needed Data

English pieces of text

1. with a high chance of being comparative
2. containing at least two known, comparable objects
   (Not like: "This is better than BMW ...")
3. understandable to many people

Objects, which are

1. comparable on at least on property
2. known by many people

Everything should be as domain unspecific as possible.

English text: Common Crawl

- CommonCrawl[1] is a freely accessible data set of crawled websites
- A preprocessed version[2] was used
    - HTML was removed
    - Splitted into sentences
    - Duplicates were removed
- 3,288,963,864 unique sentences; inserted into an Elasticsearch index
- Comparisons: 428,932 sentences contain *is better than*

[1] http://commoncrawl.org
[2] [Panchenko et al., 2018]

English text: Common Crawl

- CommonCrawl[1] is a freely accessible data set of crawled websites
- A preprocessed version[2] was used
  - HTML was removed
  - Splitted into sentences
  - Duplicates were removed
- 3,288,963,864 unique sentences; inserted into an Elasticsearch index
- Comparisons: 428,932 sentences contain *is better than*

[1] http://commoncrawl.org
[2] [Panchenko et al., 2018]

Objects and Domains

- The objects were taken from three domains
- **Computer Science**: operating systems, abstract concepts, software, ...
- **Brands**: cars, food, electronics, ...
- **Random**: book authors, soccer teams, universities, ...
- 271 pairs in total

Obtaining objects

- Wikipedia's "List of . . ." pages were used to select suitable objects for Computer Science and Brands
- Random
  - 25 seed words were randomly selected (cork, Hamster, Florida, ninja. . .)
  - JoBimText[3] was used to find the 10 most similar words for each seed word
- Each object was checked against a frequency dictionary
- Objects with a frequency of zero were removed
- For each object type (Wikipedia source page or seed word), all possible combinations were created.

[3] http://ltmaggie.informatik.uni-hamburg.de/jobimtext/

Obtaining objects

- Wikipedia's "List of ..." pages were used to select suitable objects for Computer Science and Brands
- Random
  - 25 seed words were randomly selected (cork, Hamster, Florida, ninja...)
  - JoBimText[3] was used to find the 10 most similar words for each seed word
- Each object was checked against a frequency dictionary
- Objects with a frequency of zero were removed
- For each object type (Wikipedia source page or seed word), all possible combinations were created.

[3] http://ltmaggie.informatik.uni-hamburg.de/jobimtext/

Obtaining objects

- Wikipedia's "List of ..." pages were used to select suitable objects for Computer Science and Brands
- Random
  - 25 seed words were randomly selected (cork, Hamster, Florida, ninja...)
  - JoBimText[3] was used to find the 10 most similar words for each seed word
- Each object was checked against a frequency dictionary
- Objects with a frequency of zero were removed
- For each object type (Wikipedia source page or seed word), all possible combinations were created.

[3] http://ltmaggie.informatik.uni-hamburg.de/jobimtext/

Pairs: Examples

Brands	Computer Science	Random
Microsoft vs. Apple	Java vs. Python	baseball vs. hockey
Nikon vs. Leica	Eclipse vs. Netbeans	fishing vs. swimming
Coca-Cola vs. Pepsi	OpenGL vs. Direct3D	SUV vs. minivan
Nike vs. Adidas	Integer vs. Float	Kennedy vs. Nixon
Ibuprofen vs. Advil	USB vs. Bluetooth	plastic vs. wood
Ford vs. Honda	Oracle vs. MysQL	Harvard vs. Princeton

Sentence Sampling

- 21 words (like better, worse, slower, inferior, cooler) were selected as comparison **cue words**
- for 90 percent of the pairs, the index was queried for sentences containing both objects and at least one cue word
- for the remaining 10 percent, the cue word was omitted
- 2500 sentences for each domain were randomly sampled from the result

Sentence Sampling

- 21 words (like better, worse, slower, inferior, cooler) were selected as comparison **cue words**
- for 90 percent of the pairs, the index was queried for sentences containing both objects and at least one cue word
- for the remaining 10 percent, the cue word was omitted
- 2500 sentences for each domain were randomly sampled from the result

Sentence Sampling

- 21 words (like better, worse, slower, inferior, cooler) were selected as comparison **cue words**
- for 90 percent of the pairs, the index was queried for sentences containing both objects and at least one cue word
- for the remaining 10 percent, the cue word was omitted
- 2500 sentences for each domain were randomly sampled from the result

Sentence Sampling: Examples

1. “There is no doubt Python is better than Ruby at any in aspect you will pick.”
2. “Goodnight NetBeans, Hello Eclipse”
3. “stone is harder than metal”
4. “arrrggghh...Python is a terrible language - only Perl sucks worse.”
5. “Good to see again a Renault ahead of a Ferrari.”

Crowdsourcing

Task Design

- All sentences were annotated via the crowdsourcing platform Crowdflower[4].
- A prestudy was conducted to assess the quality of the annotation guidelines and the sentence selection process.
- In the prestudy, about 25 percent were labeled as comparative.
- Each sentence was annotated by at least five annotators.

[4] https://crowdflower.com

Task Design: Problems

Initially, the annotators were asked to answer the question:

People only believe your drive a **BMW:[OBJECT_A]** is because you are a wealthy individual who can afford a better car than a **Honda:[OBJECT_B]** Civic.

What describes the comparison in the sentence above best? (required)

- ◯ The first object is BETTER than the second object. (BETTER)
- ◯ The first object is WORSE than the second object. (WORSE)
- ◯ The sentence is comparative, but neither BETTER or WORSE fit. (OTHER)
- ◯ There is no comparison. (NONE)

Task Design: Problems

- People confused `OTHER` with `NONE` frequently
- People were dissatisfied because the distinction was to hard
- `OTHER` and `NONE` were hardly distinguishable in first classification experiments
- After 750 annotated sentences per domain, `OTHER` was dropped
- `OTHER` was merged into `NONE` for the classification experiments

Task Design: Problems

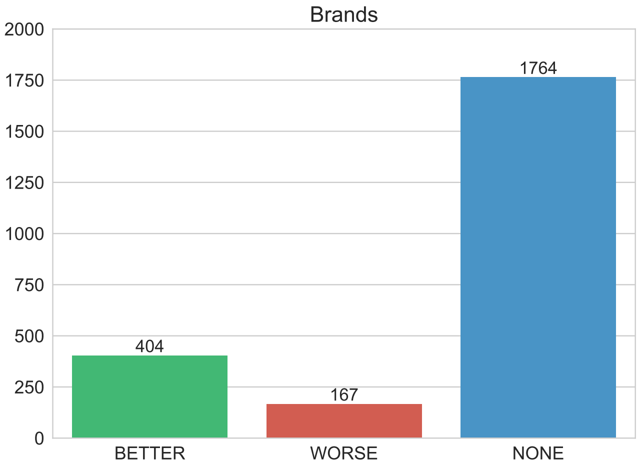
- People confused `OTHER` with `NONE` frequently
- People were dissatisfied because the distinction was to hard
- `OTHER` and `NONE` were hardly distinguishable in first classification experiments
- After 750 annotated sentences per domain, `OTHER` was dropped
- `OTHER` was merged into `NONE` for the classification experiments

Task Design: Problems

- The annotation guidelines stated that all questions should be labelled as `NONE`.
  (For instance, "Is Python better than Ruby?")
- This was frequently overlooked by the annotators.

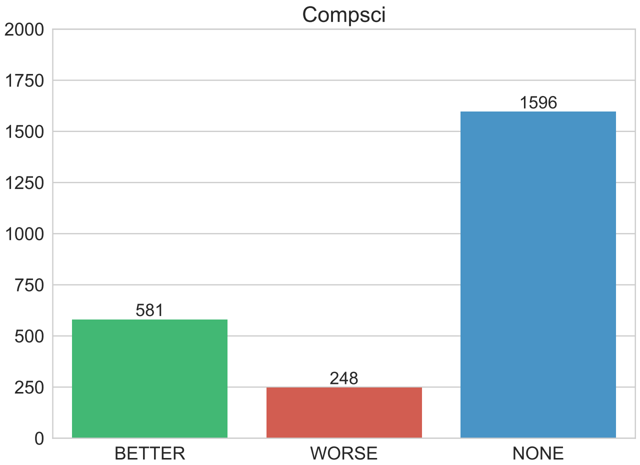
Results: Brands

- 2335 sentences in total
- 571 comparative sentences (24 percent)



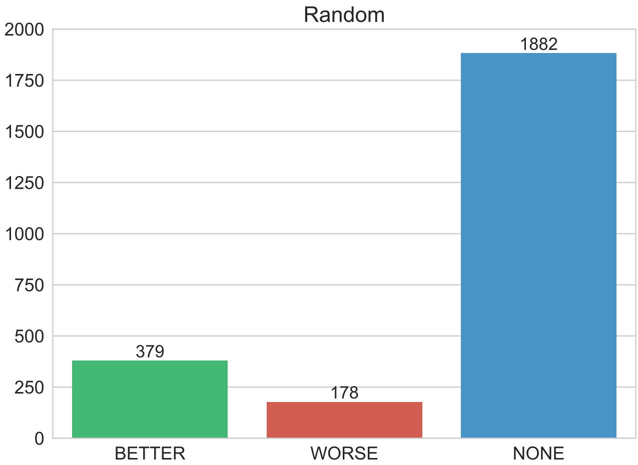
Results: Computer Science

- 2425 sentences in total
- 829 comparative sentences (34 percent)



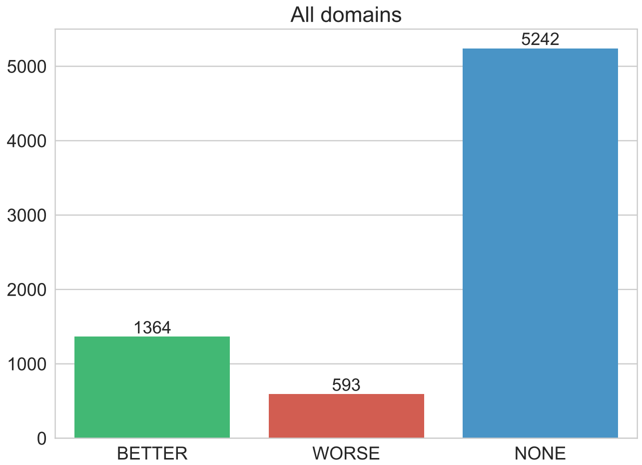
Results: Random

- 2439 sentences in total
- 557 comparative sentences (22 percent)



Results: All Domains

- 7199 sentences in total
- 1957 comparative sentences (27 percent)
- the class BETTER is more than two times bigger than WORSE



Results: All Domains

- 7199 sentences in total
- 1957 comparative sentences (27 percent)
- the class `BETTER` is more than two times bigger than `WORSE`

Annotation confidence for all domains. The confidence is calculated as judgments for majority class/total judgments.

Confidence	Sentences	% of data set
100%	5111	71.00
91-99%	0	0.00
81-90%	75	1.04
71-80%	1057	14.68
61-70%	33	0.46
51-60%	754	10.47
0-50%	169	2.35

Classification

Setup

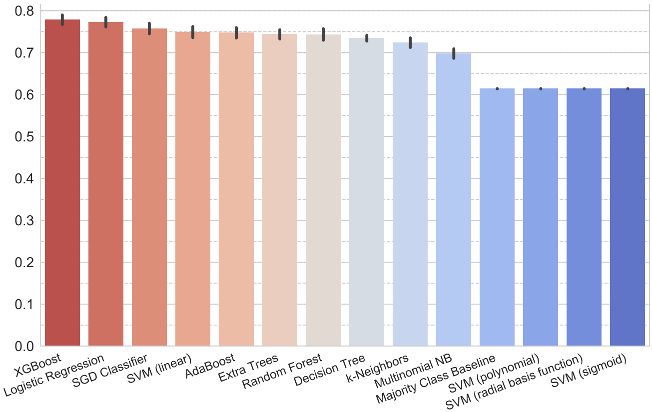
- The 7199 sentences were split into a development (5759) and held-out (1440) set.
- All experiments were conducted on the development set and evalutated with k-folds cross validation ($k = 5$)
- Two setups:
  - **Three classes**: `NONE`, `BETTER` and `WORSE`
  - **Binary**: `NONE` and `ARG` ($=$ `BETTER` $\cup$ `WORSE`)

Setup

- The 7199 sentences were split into a development (5759) and held-out (1440) set.
- All experiments were conducted on the development set and evalutated with k-folds cross validation ($k = 5$)
- Two setups:
  - **Three classes**: `NONE`, `BETTER` and `WORSE`
  - **Binary**: `NONE` and `ARG` ($=$ `BETTER` $\cup$ `WORSE`)

Algorithms

- 13 classification algorithms were tested with a bag-of-words-model
- XGBoost with 1000 base estimators was used in all experiments (gradient boosted decision trees; presented in [Chen and Guestrin, 2016])
- The graphic shows the f1 score and standard derivation (black bar).



Baseline: Three classes

Random (stratified) baseline

	precision	recall	f1 score
B	0.19 ±0.01	0.21 ±0.01	0.20 ±0.01
W	0.06 ±0.02	0.05 ±0.02	0.06 ±0.03
N	0.73 ±0.00	0.73 ±0.00	0.73 ±0.00
avg.	0.57 ±0.00	0.58 ±0.01	0.57 ±0.00

Most frequent class baseline

	precision	recall	f1 score
B	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
W	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
N	0.73 ±0.00	1.00 ±0.00	0.84 ±0.00
avg.	0.53 ±0.00	0.73 ±0.00	**0.61** ±0.00

B = BETTER, W = WORSE, N = NONE,

Baseline: Binary

Random (stratified) baseline

	precision	recall	f1 score
ARG	0.26 ±0.03	0.26 ±0.03	0.26 ±0.03
N	0.72 ±0.01	0.72 ±0.01	0.72 ±0.01
avg.	0.60 ±0.02	0.60 ±0.02	0.60 ±0.02

Most frequent class baseline

	precision	recall	f1 score
ARG	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
N	0.73 ±0.00	1.00 ±0.00	0.84 ±0.00
avg.	0.53 ±0.00	0.73 ±0.00	**0.61** ±0.00

ARG = BETTER + WORSE, N = NONE

Features

Feature Overview

- Bag-of-words
- 500 most frequent part-of-speech bi-, tri and four-grams
- Mean word embedding vector (GloVe vectors, size 300)
- Boolean feature capturing the appearance of a comparative adjective (Contains JJR)
- Sentence Embeddings
- Dependency Paths

Sentence Embeddings

- Dense vector representation for phrases, similar to word embeddings
- Several approaches, for instance SkipThrough [Kiros et al., 2015], Paragraph Vectors [Le and Mikolov, 2014] and **InferSent** [Conneau et al., 2017]
- A pretrained InferSent model[5] was used in the thesis

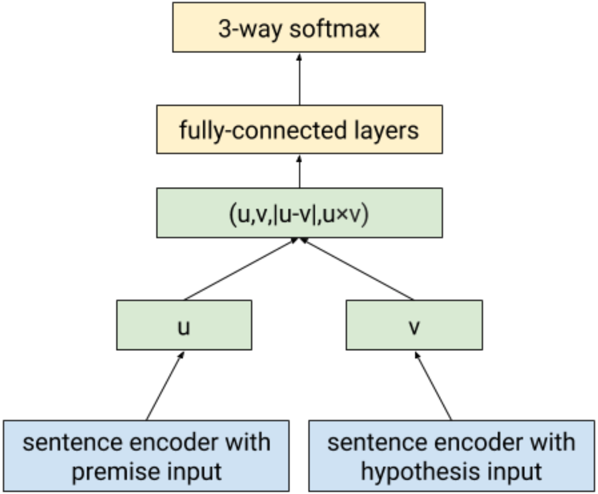
[5] https://github.com/facebookresearch/InferSent

Sentence Embeddings

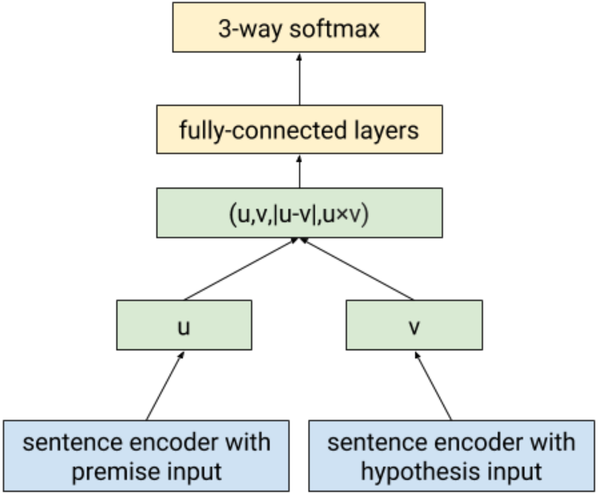
- Dense vector representation for phrases, similar to word embeddings
- Several approaches, for instance SkipThrough [Kiros et al., 2015], Paragraph Vectors [Le and Mikolov, 2014] and **InferSent** [Conneau et al., 2017]
- A pretrained InferSent model[5] was used in the thesis

[5] https://github.com/facebookresearch/InferSent

- Neural Network trained on the Stanford Natural Language Inference (SNLI) corpus
- SNLI contains 570k sentence pairs labelled as contradiction, entailment or neutral
- BiLSTM with max-pooling and 4096 neurons as encoders
- tested on a wide range of tasks, outperforms SkipThrough and Paragraph Vectors



- Neural Network trained on the Stanford Natural Language Inference (SNLI) corpus
- SNLI contains 570k sentence pairs labelled as contradiction, entailment or neutral
- BiLSTM with max-pooling and 4096 neurons as encoders
- tested on a wide range of tasks, outperforms SkipThrough and Paragraph Vectors



HypeNet and LexNet

- HypeNet[6] combines word embeddings and (dependency) path-based information to check if two words are hypernyms
- LexNet[7] is a generalisation of HypeNet to find multiple semantic relations
- HypeNet creates a string representation of the dependency path between two words
- the string representations are then encoded using an LSTM
- the average of all paths for each word pair is used as the path feature

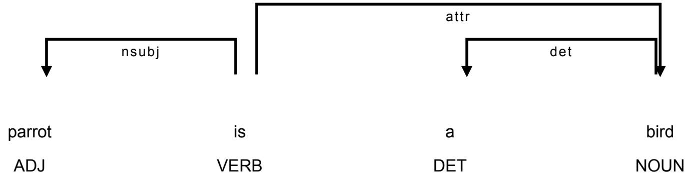
[6] [Shwartz et al., 2016]
[7] [Shwartz and Dagan, 2016]

HypeNet and LexNet

- HypeNet[6] combines word embeddings and (dependency) path-based information to check if two words are hypernyms
- LexNet[7] is a generalisation of HypeNet to find multiple semantic relations
- HypeNet creates a string representation of the dependency path between two words
- the string representations are then encoded using an LSTM
- the average of all paths for each word pair is used as the path feature

[6] [Shwartz et al., 2016]
[7] [Shwartz and Dagan, 2016]

HypeNet and LexNet: Example



- `X/NOUN/nsubj/< be/VERB/ROOT/- Y/NOUN/attr/>`
- Each node contains lemma, part of speech, dependency label and the edge direction
- Expectation: path embeddings add valuable information to sentence embeddings

HypeNet and LexNet: Features

- Two features based on HypeNet paths:
- **LexNet (original)** creates paths as described in the paper
  - maximum length of four
  - the first object must be reachable from following only left edges, starting from the lowest common head
  - the second object must be reachable from following only right edges
  - 1519 sentences without a path
- **LexNet (optimized)**
  - maximum length of sixteen
  - no restrictions on the direction
  - 399 sentences without a path

HypeNet and LexNet: Features

- Two features based on HypeNet paths:
- **LexNet (original)** creates paths as described in the paper
  - maximum length of four
  - the first object must be reachable from following only left edges, starting from the lowest common head
  - the second object must be reachable from following only right edges
  - 1519 sentences without a path
- **LexNet (optimized)**
  - maximum length of sixteen
  - no restrictions on the direction
  - 399 sentences without a path

HypeNet and LexNet: Features

- Two features based on HypeNet paths:
- **LexNet (original)** creates paths as described in the paper
  - maximum length of four
  - the first object must be reachable from following only left edges, starting from the lowest common head
  - the second object must be reachable from following only right edges
  - 1519 sentences without a path
- **LexNet (optimized)**
  - maximum length of sixteen
  - no restrictions on the direction
  - 399 sentences without a path

Preprocessing

Selection of the sentence part

- the whole sentence
- all words between the first and the second object
- all words before the first object
- all words after the second object

Object replacement

- leave the objects
- remove both objects
- replace both objects with the term `OBJECT`
- replace the first object with `OBJECT_A` and the second with `OBJECT_B`

Preprocessing

Selection of the sentence part

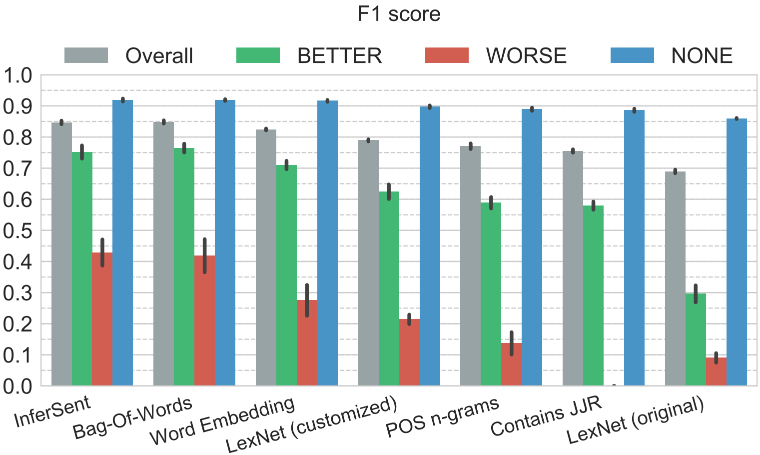
- the whole sentence
- all words between the first and the second object
- all words before the first object
- all words after the second object

Object replacement

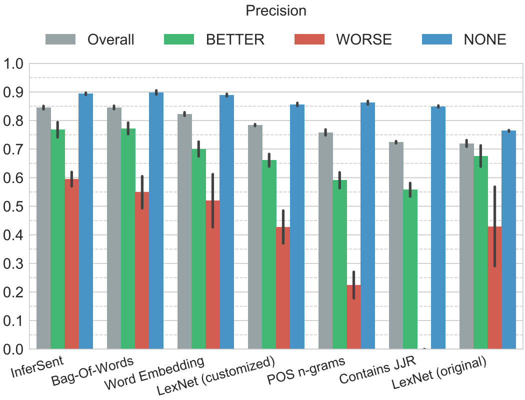
- leave the objects
- remove both objects
- replace both objects with the term `OBJECT`
- replace the first object with `OBJECT_A` and the second with `OBJECT_B`

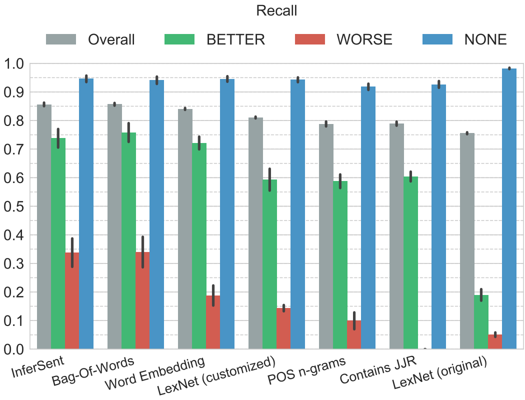
Training Results

Three classes: F1 score

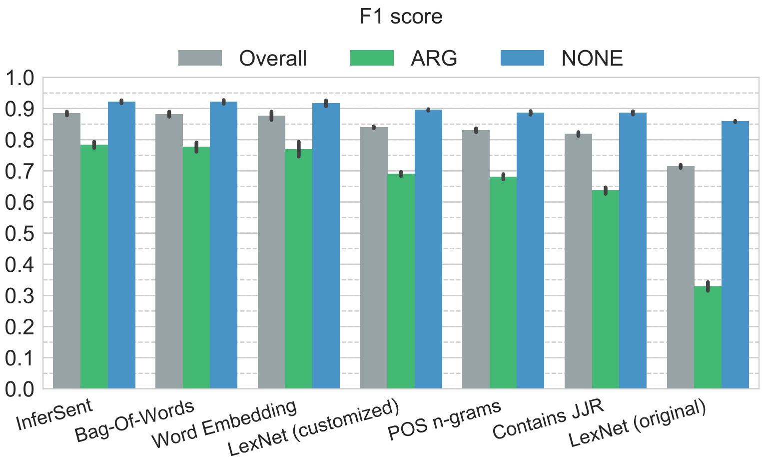


Three classes: Precision and Recall

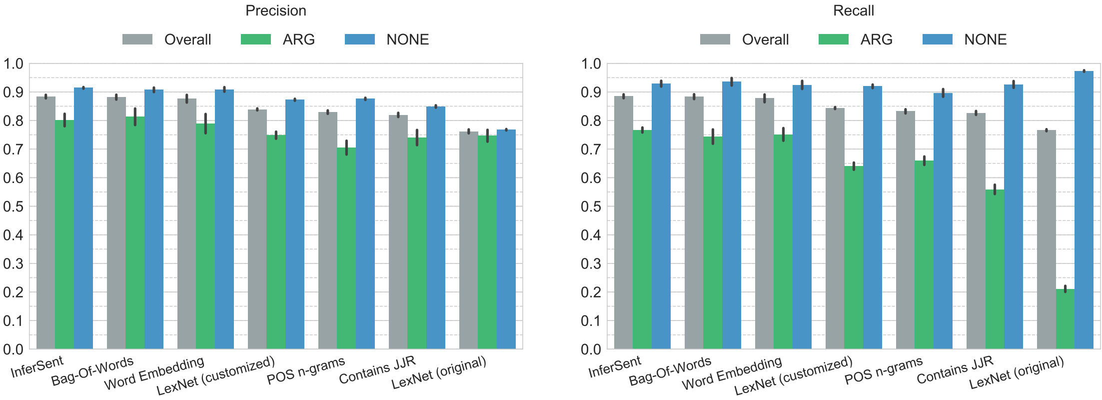




Binary: F1 score



Binary: Precision and Recall



Intermediate Results

- As expected, `WORSE` is hard to recognize
- InferSent, Bag-Of-Words and Mean Word Embeddings have a similar f1 score
- InferSent is more precise on `WORSE`
- The best f1 score is 24 points above the baseline.
- The original LexNet setup the worst, but still above the baseline.
- The binary scenario is only slightly better than the three class scenario
- Using only the middle part of the sentence **increases the f1 score by 6-13 points**.
- The objects contribute only little to the result; removing or replacing did not alter the f1 score by more than 0.005 points.

Intermediate Results

- As expected, `WORSE` is hard to recognize
- InferSent, Bag-Of-Words and Mean Word Embeddings have a similar f1 score
- InferSent is more precise on `WORSE`
- The best f1 score is 24 points above the baseline.
- The original LexNet setup the worst, but still above the baseline.
- The binary scenario is only slightly better than the three class scenario
- Using only the middle part of the sentence **increases the f1 score by 6-13 points**.
- The objects contribute only little to the result; removing or replacing did not alter the f1 score by more than 0.005 points.

Intermediate Results

- As expected, `WORSE` is hard to recognize
- InferSent, Bag-Of-Words and Mean Word Embeddings have a similar f1 score
- InferSent is more precise on `WORSE`
- The best f1 score is 24 points above the baseline.
- The original LexNet setup the worst, but still above the baseline.
- The binary scenario is only slightly better than the three class scenario
- Using only the middle part of the sentence **increases the f1 score by 6-13 points**.
- The objects contribute only little to the result; removing or replacing did not alter the f1 score by more than 0.005 points.

Error analysis 1

Errors made by InferSent and LexNet (optimized)

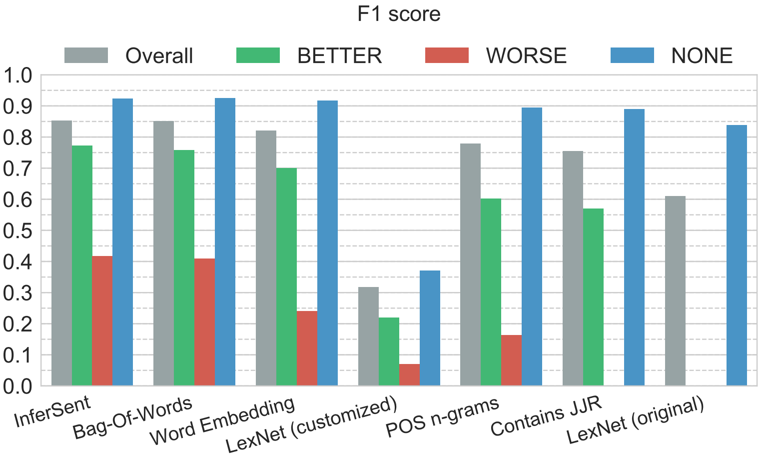
- 1311 sentences were incorrectly classified (three class scenario).
- 607 errors were made by both features.
- 220 additional were exclusively made by InferSent, 484 by LexNet.
- the errors made in the binary scenario are similar (1183 errors, 739 shared with the three-class scenario, 444 new).

Error analysis 2

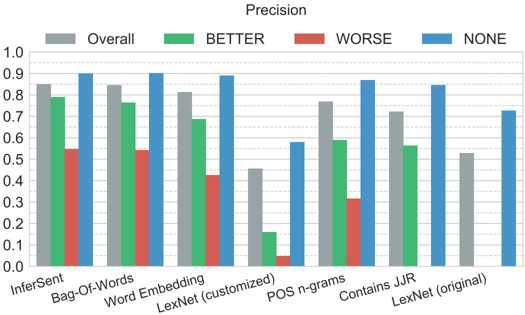
- The majority of errors was made on sentences with a high annotation confidence.
- Identified problems:
  - questions
  - negations
  - missing cue words
  - comparative sentences which do not compare the objects
  - missing context / knowledge
- `WORSE` was confused with `NONE` more often than with `BETTER`

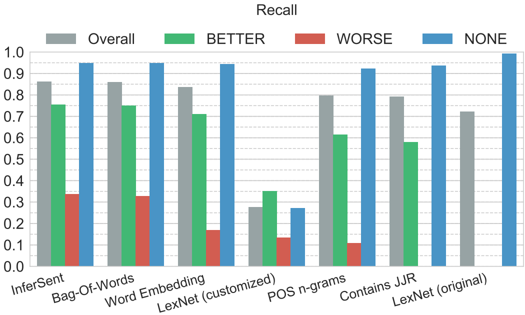
Evaluation with the held-out data

Three classes: F1 score

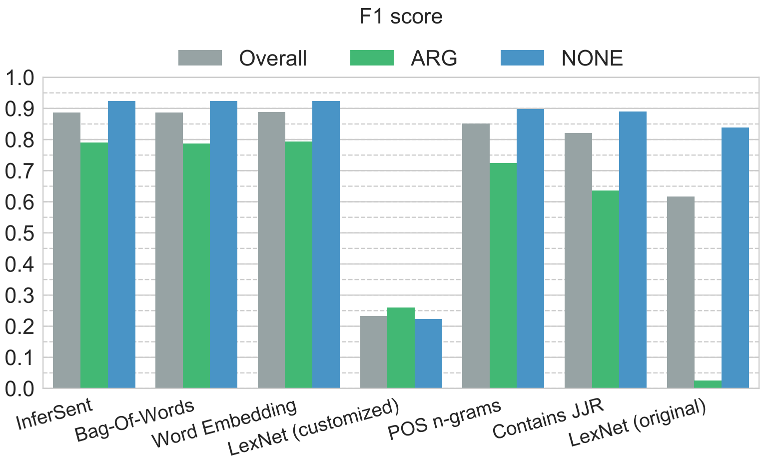


Three classes: Precision and Recall

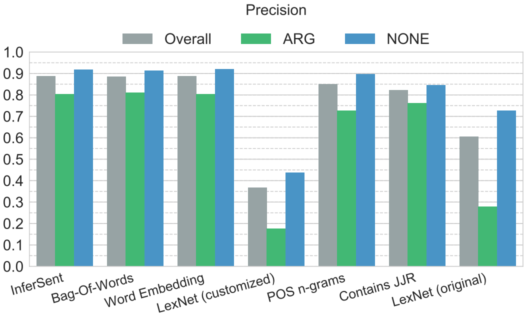


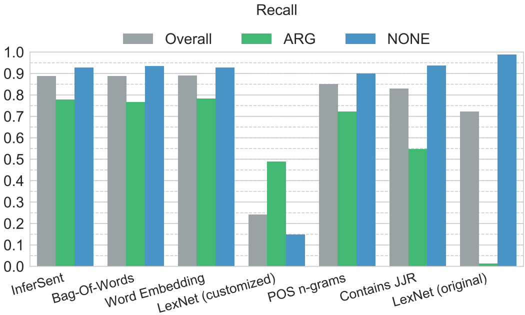


Binary: F1 score



Binary: Precision and Recall





Results

- InferSent is again the best feature
- The LexNet feature did not generalize
  - 2344 unique paths for 5759 sentences (training set)
  - 594 unique paths for 1441 sentences (held out set)
  - training and held had only 81 paths in common
- No feature combination was better than InferSent

Results

- InferSent is again the best feature
- The LexNet feature did not generalize
  - 2344 unique paths for 5759 sentences (training set)
  - 594 unique paths for 1441 sentences (held out set)
  - training and held had only 81 paths in common
- No feature combination was better than InferSent

Conclusion and Future Work

Conclusion

- The best feature could yield an f1 score of 0.85; 24 points **above** the baseline.
- Simple features (bag-of-words) perform almost equal to more complex features.
- HypeNet needs way more training data!
- The objects are not important for the classification at all.
- Preprocessing is crucial to achieve good scores.
- Contrary to the expectations `WORSE` is more similar to `NONE` than to `BETTER`.
- All in all, the crowd sourcing and classification worked satisfactory.

Conclusion

- The best feature could yield an f1 score of 0.85; 24 points **above** the baseline.
- Simple features (bag-of-words) perform almost equal to more complex features.
- HypeNet needs way more training data!
- The objects are not important for the classification at all.
- Preprocessing is crucial to achieve good scores.
- Contrary to the expectations `WORSE` is more similar to `NONE` than to `BETTER`.
- All in all, the crowd sourcing and classification worked satisfactory.

Conclusion

- The best feature could yield an f1 score of 0.85; 24 points **above** the baseline.
- Simple features (bag-of-words) perform almost equal to more complex features.
- HypeNet needs way more training data!
- The objects are not important for the classification at all.
- Preprocessing is crucial to achieve good scores.
- Contrary to the expectations `WORSE` is more similar to `NONE` than to `BETTER`.
- All in all, the crowd sourcing and classification worked satisfactory.

Future work

- More data!
- Add more features to capture special cases like questions
- Use surrounding sentences for context information and coreference resolution
- Test in a real world application

References I

Chen, T. and Guestrin, C. (2016).
Xgboost: A scalable tree boosting system.
In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. Association for Computing Machinery.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017).
Supervised learning of universal sentence representations from natural language inference data.
In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

References II

Fiszman, M., Demner-Fushman, D., Lang, F. M., Goetz, P., and Rindflesch, T. C. (2007).
Interpreting comparative constructions in biomedical text.
In *Biological, translational, and clinical language processing, BioNLP@ACL 2007, Prague, Czech Republic, June 29, 2007*, pages 137–144, Prague, Czech Republic. Association for Computational Linguistics, Association for Computational Linguistics.

References III

Gupta, S., Mahmood, A. S. M. A., Ross, K., Wu, C. H., and Vijay-Shanker, K. (2017).
Identifying comparative structures in biomedical text.
In *BioNLP 2017, Vancouver, Canada, August 4, 2017*, pages 206–215, Vancouver, Canada. Association for Computational Linguistics.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015).
Skip-thought vectors.
In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12,*

References IV

2015, Montreal, Quebec, Canada, pages 3294–3302. Neural Information Processing Systems Conference.

Le, Q. V. and Mikolov, T. (2014).
Distributed representations of sentences and documents.

Panchenko, A., Ruppert, E., Faralli, S., Ponzetto, S. P., and Biemann, C. (2018).
Building a web-scale dependency-parsed corpus from commoncrawl.
In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*, Miyazaki, Japan. European Language Resources Association.

References V

Park, D. H. and Blake, C. (2012).
Identifying comparative claim sentences in full-text scientific articles.
In *Proceedings of the Workshop on Detecting Structure in Scholarly Discourse*, ACL '12, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shwartz, V. and Dagan, I. (2016).
The roles of path-based and distributional information in recognizing lexical semantic relations.
CoRR, abs/1608.05014.

References VI

Shwartz, V., Goldberg, Y., and Dagan, I. (2016).
Improving hypernymy detection with an integrated path-based and distributional method.
In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, volume 1.

Thank you! Questions?

franzek@posteo.net