Text and Multimedia Mining Summary

Nadezhda Dobreva

September 2, 2023

1 Week 1

1.1 Basics

- Text and multimedia mining: acquisition of knowledge from unstructured text and multimedia data;
- There are many different types of data – you can do multimedia/text mining for any field. It's important that it's unstructured: work with natural language, not databases, records;
- The knowledge we try to obtain concerns the data and the data we're trying to represent, not the method of acquisition (i.e. the algorithm). Knowledge is defined as "reusable information" which we can use across different situations;
- TxM data reflects its creators, experience, the world, language use;
- Main principles/limitations:
  - Limitations of ethics: shouldn't harm the people who created the data or those it's about;
  - Limitations of the approach: you can't get perfect knowledge; the strength of your project rests on your ability to recognize and discuss the limitations that you can't tackle.
- TxMM uses machine learning but isn't machine learning!
- Data mining needs structured data for extracting patters and relations in it, while TxMM uses unstructured data – they are not the same!
- There are 4 dimensions to any research project for TxMM: question (the knowledge needed), data (how much, from where?), approach (what kind of mining?), validation.

1.2 Validation Approaches

- Internal validity: setting up the approach correctly, not making any unjustified/hidden assumptions, no mistakes in the implementation;
- External validity: should answer an open question (so, no spurious correlations or playing around with some random data for no reason), the collection of data should be appropriate and representative, the output is confirmed by experiments, observations or human judgement (or express as limitation if you can't confirm your results);
- When making conclusions, keep in mind that the data you're using probably doesn't cover all possible sources and so, you're probably not analyzing the general population but e.g. people with strong digital presence;
- Without connection to reality (external validity), the results are worthless.

1.3 Textual Data

- Text is unstructured, comes from mixed sources (can be multi-style, multi-lingual), noisy (spelling errors, OCR noise);
- Just like language, it is infinite (vocabulary can be increased infinitely, words come to existence all the time), interconnected, interpreted in context;
- Zipf's law: the frequency of any word is inversely proportional to its rank in the frequency table.

2 Week 2

2.1 Some Definitions

- Natural Language Processing – processing of text (natural language) by a computer;
- Computational linguistics – the field of study that makes computational models for linguistics;
- Text mining - extracting knowledge from text – the focus of the course is on this;
- The first phase of the text mining process: collecting data in the form of raw text → clean, filter and convert the raw text into clean text → preprocess the clean text with NLP tools to obtain a linguistically annotated text → select and create features of it → obtain the product: a feature vector.

2.2 Raw Text

- Raw material, written so we have digitized documents;
- If the data is spoken (in the form of audio/video) you have to transcribe it to get textual data;
- Born-digital documents: created on a computer (text, html, word, pdf, etc.). Nowadays most are like this;
- OCR (optimal capture recognition):
  - A program that looks at a picture and converts the characters it sees into text. There are some limitations to it, however;
  - Information loss: spelling errors that lead to nonsense words or deleting of words that weren't recognized;
  - Fabrication: insertion words that weren't originally in the text.

2.3 Cleaning the Text

- The input text can contain photos, tables, graphics, design info, ads, copyright statements, etc. (especially web pages). html pages also have lists of links, navigation bars, etc. All of this should be cleaned up;
- storing the clean text:
  - Character encoding: the way the computer displays a file so that a human can understand it. One of the most important aspects of computers in general (translating binary strings to characters). ASCII – 7 bit encoding;
  - It used to be that every language has its own encoding, which is efficient for 1 language but incompatible with others; differing operating systems and software lead to problems;

  - Unicode character set: universal standard for all writing systems, independent of platform, software, vendor. It's a description of how the character looks like. UTF-8 is the most common encoding of Unicode (compatible with ASCII as well);
  - Markup: annotating the file with extra info about the text and its context – metadata (origin, location, genre, etc. and also indications of how a text should look like (newlines, spaces, letter type and size, bold or italics, etc.). It's a set of conventions used together for encoding text. It defines what is allowed, required, the format and its meaning;
  - html – a markup format for web browsers that define how to display a text. XML is intended to store or transport data. Both of them are Descriptive: they assign boundaries and a name to a piece of text. And the actual processing is defined outside text: XML and HTML don't DO anything, it's just a description within tags so you need a program or a human to do anything with the info in the tags.
- Clean text: after digitalization, data conversion and cleaning we get actual textual content separated from other info, uniformly represented by appropriate character encoding but can still contain spelling errors, unconventional spelling or emoji.

2.4 Processing with NLP tools

- We go from strings to some knowledge representation by extracting information from the text: splitting into words, making POS tags, finding syntactic structures and entities and relations, trying to find logic within it;
- Tokenization:
  - Definition: recognition of word boundaries; converting the text from a sequence of characters to a sequence of word tokens. This is a basic step before doing any other linguistic annotation;
  - Usually it is done with rules (regex) and is relatively easy since words are separated by spaces (that's the simplest approach: split the text on whitespace characters);
  - A language-specific problem;
  - Dictionary:
    * Word = a delimited string of characters;
    * Whitespace = space, tab, newline;
    * Token = an instance of a word occurring in a document;
    * Type = an equivalence class of tokens (works like a dictionary);
    * Delimiter = whitespace and possibly sentence marking punctuation.
  - The exact definition of tokens and delimiters is application-dependent;
  - The type/token ratio is a characteristics of a document or collection;
  - Not too easy of a task since you need to make some decisions like what to do with numerical data, should you parse hyphens, should you remove tags, hashtags, emojis. There are also some problematic things like compound words (should we split into separate words?), multi-word expressions (shouldn't be split because they'll lose their meaning), etc.
- Sentence splitting: easy to do for Western languages since we can use a rule-based approach (start with capital letter and end with a . ? !). There are also hard cases like names that include punctuation, abbreviations, etc;

- Part of speech (POS) tags:
  - Assigning each word in the corpus its grammatical label in its context. POS tags refer to the syntactic role of each word in a sentence;
  - POS tags are morphosyntactic categories: they describe the words in terms of how they pattern together and how they are internally constructed (e.g. what suffixes and prefixes they have);
  - Universal dependencies (UD) project: a tagset which is meant to capture world classes across as many languages as possible:
    * Open class tags (content words): noun, verb, adjective, adverb, proper nouns and interjections;
    * Closed class tags (function words): they have little lexical meaning of their own but rather help organize the components of a sentence. Adpositions, auxiliary verbs, conjunctions, pronouns, determiners, numerals, particles;
    * Other: punctuation and symbols.
  - There's a POS-tagger which automatically predicts labels with 98% accuracy;
  - It's the most commonly used type of linguistic annotation, it precedes other steps of more complicated annotation;
  - It allows us to count the most frequent nouns, verbs, etc., or determine what kind of nouns are associated with what kind of verbs. There's even more information: verbs have tenses and aspects, nouns can have numbers, adjectives – degrees, etc. (morphosyntactic attributes);
  - Representing text as both words and POS tags enriches the representation and enables a deeper more principled analysis;
  - POS tagging is considered solved but only half of sentences are flawless, there's bad transfer to other text genres and corpora and some disambiguation is often at ¡90%.
- Base word forms achieved after text normalization:
  - Definition: string transformations that remove irrelevant distinctions for downstream applications;
  - Assign words in context their basic word form or meaning. Word disambiguation is the problem of identifying the intended sense of each word in a document. We can use lemma or root/stem of the word;
  - Case conversion: Sentence-initial capitalization is removed, but not all capitalization – sometimes it's necessary;
  - Can be useful to convert all numbers and dates to special tokens, depending on application;
  - Historical texts spelling variations can be normalized to concurrent standard form. Excessive lengthening can be shortened;
  - Lemma is a canonical form of a word that stands in for a set of inflected words. The sense is a property of the lemma, not the word. We need to identify the correct lemma and then choose the correct sense from the inventory associated with it;
  - Eliminating the inflectional affixes (-s or -ing or -ed): done by a stemmer. Root/stem is the portion of a word that is common to a set of forms when all affixes are removed and isn't further analyzable into meaningful elements. It is morphologically simple and

carries the principle portion of meaning of the words in which it functions. So the stem is the part of the word that never changes;

  - Lexical semantic relations:
    * Synsets are groups of lemmas (or phrases) that are synonymous. A lemma is polysemous if it participates in multiple synsets;
    * Antonymy: x is opposite of y;
    * Hyponymy: x is a special case of y;
    * Meronymy: x is a part of y;
    * Classification of those can be done using characteristic patterns between words (but, such as, etc).
  - Normalization reduces the size of the feature space which can help in generalization. Another way to do that, is removing stop words: words that have little contribution to the content and occur frequently (about, on, and, in, etc). We create a stoplist of words that are to be ignored.

- Syntactic parsing:
  - Analyzing a sentence in its syntactic constituents;
  - The main aspects are the type of grammar, type of syntactic analysis (dependency vs constituency), grammar induction, depth of analysis (full/partial parsing);
  - Grammar types:
    * Symbolic: rules that code grammatical relations;
    * Probabilistic: rules extracted from a corpus, using statistical info to resolve ambiguity;
    * Constraint-based: describes what is not allowed by the grammar.
  - Type of syntactic analysis:
    * Constituency: a word or group of words which is a single unit within a hierarchical structure. Each element is connected to one or more elements. The rules are terminal and nonterminal;
    * Dependency: minimal representation. The verb is the center of the function and each element is connected to one other element in a relation. One element is the head and the other is the dependent – it's a flatter structure;
    * Dependency grammar is better suited for languages in which word order doesn't matter.

- Named entity recognition: the core entity types are people, locations, organizations, dates, times, percentages. Can be done using BIO notation: each token at the beginning of a name is labeled with a B-prefix, each token within a name span is labeled with an I-prefix, and tokens that aren't part of a name have the O-prefix. Can also have special labels for the Last tokens from the name, or Unique tokens in name spans;
- Going further, we can identify relations, logic predicates, speech acts. Deeper and deeper the representations get less and less robust (although more useful). They require more sophisticated NLP techniques and more human effort. There's a tradeoff between doing a deeper analysis that might have errors but will give us direct knowledge extracted from the text, and doing shallow analysis that's more robust but won't give us as deep knowledge;

- Treebank: corpus annotated fully with syntactic information. Automatic parsing is possible for its creation but it would need a round of manual correction;
- Semantic concordance: a corpus in which each open-class word is tagged with its word sense from the target dictionary/thesaurus;
- Supervised learning is only possible if there are enough labeled examples. Each polysemous lemma requires its own training set! This is difficult, so we use usupervised and semi-supervised methods.
- Referring expressions: linguistic expressions like names and pronouns refer to certain referents (entities, events in the real world, other linguistic expressions that occurred previously in the text). To understand what the text is about you need to understand those references.

2.5 Feature Creation

- Happens from the linguistically annotated text;
- We want to make a representation that at computer can understand;
- Bag of words model: a dictionary (unordered list) of all words in a document collection. The simplest representation is sequence of word/binary vector (documenting the presence of a term). Can also have a weighted vector with one real-valued weight for each term or the counts of each term;
- The choice of feature representation is task, data and model dependent;
- It is the case that binary representations outperform vectors based on counts. One explanation could be that words tend to appear in clumps: if a word appeared once in a document, it's likely to appear again.

3 Week 3

3.1 Types of Tasks

- Classification;
- Clustering;
- Topic modeling;
- Ranking;
- Representation creation;
- Word embeddings.

3.2 Machine Learning

- Supervised learning:
  - A computer learns how to compute a function based on a set of examples of the input value (training data) and the corresponding expected outcome. Training data is used for the building of the model and testing data is used for its evaluation;
  - Typically the outcomes are provided by the humans;
  - Once the function is learned the computer would be able to take unseen inputs and compute the function on them;
  - Assigning the correct "label" to an input is called a classification task;

  - The input is often represented as a feature vector and the computer learns and optimal way to combine these features with weights to indicate their importance and influence on the final outcome.

- Unsupervised learning:
  - We only have the inputs without knowing the outcomes. No humans are involved;
  - We can't really learn how to compute the function, but we can learn latent properties or structures of the inputs;
  - The computer can learn that some data instances are very similar (can be grouped together) and the whole dataset can be represented by a number of clusters – a clustering task.

3.3 Classification

- The input is a data point (text document, image) and the output is a class label for that data point. The classes (categories) are pre-defined in advance. The model can do it because it was trained on data point - label pairs;
- Text categorization helps us enrich text representation (i.e. achieve more understanding of text) and infer properties of entities associated with text data;
- Binary classifiers can predict one of two labels, multi-class classifier assigns multiple labels;
- There is human involvement in data collection and preprocessing: text search, tokenization of words, selective lowercasing and deaccenting, etc;
- Three categories:
  - Instance-based classifiers / lazy learners: don't model the class labels explicitly but compare the new instances with instances seen before using a similarity measure. Most calculation is performed during testing time;
  - Generative classifiers: model the data distribution of each class and assign labels based on the likelihood of the object being observed according to the distributions;
  - Discriminative classifiers: compute features of a text object that can provide a clue about which category the object should be in, and combine them using weights.
- Steps:
  - Define the task: what is the text input (complete documents, sections, paragraphs, sentences)? What categories (topics ones, sentiment ones, recommendation, etc.)?
  - Collect the training data:
    * Can use existing labelled data (labeled by someone else, or has naturally occurring labels; e.g. hashtags in twitter) or manually label data ourselves (or ask domain specialists to do it for us);
    * The manual approach can only work if the categories are very clearly defined and easily distinguishable based on surface features in the text, and there is sufficient domain knowledge. There are downsides to this, however: labour-intensive, may not be possible to come up with completely reliable rules, and they might be inconsistent with each other;
    * The more examples the better: at least dozens/hundreds per category. The more difficult the problem the more examples needed;
    * Judging the quality of the labels: the human labelled reference data is the "gold standard." However, two human annotators never fully agree, so we always must

have at least part of the example data labelled by multiple annotators. The inter-rater agreement is a measure of the reliability of the labels;

  - Cohen's Kappa: a formula used to calculate the agreement between two annotators.

- Preprocess the data;
- Extract features. Usually most features aren't useful and the bulk of the decision is based on a small subset of the features – determining this subset is called feature selection. Some features are unigrams, bigrams, etc;
- Train the classifier:
  - Can use Naive Bayes (generative classifier), Decision Trees, Neural Networks, Support Vector Machines, kNN classifier;
  - An SVM finds the hyperplane that maximizes the margin (space) between two classes. The vectors that define the hyperplane are the support vectors;
  - k-nearest neighbor classifier: plot the collection of labeled vectors, plot the test input vector with unknown label. Find the k nearest neighbors and count which class has more members among them. This method is sensitive to the similarity function used to calculate the neighbors and to k which means we should do a process of hyperparameter selection to find out what k to use. If there happen to be some outliers closeby to the new sample, it will be incorrectly classified. A lazy learner (no explicit training step). Can be applied to any distance measure and any document representation;
  - Nearest mean classification: represent the classes using the class average (the mean of the data points with that class label) – the mean is taken with respect to each dimension individually. Plot the input and compute which class is closest. This doesn't have the bias kNN does;
  - Perceptron: single-layer NN which implements a binary linear classifier. It draws a linear boundary between two classes. If the perceptron "fires" when it's given some input then that class of that input is "positive";
  - Using development data set to evaluate how the training is going is good to prevent overfitting, or losing generality. A trained model is robust if it isn't prone to overfitting;
- Evaluation:
  - We need to evaluate the classifier to know whether it's doing what we think/want it to be doing and whether it's performing well in terms of user satisfaction, accuracy, speed, system reliability;
  - Training and test sets are mutually exclusive for fair evaluation and to prevent overfitting. Typically a 80-20 split; beware of the split so that you don't create some kind of pattern that will hurdle the learning/performance;
  - Validating the model using a validation data set is important. It's a held-out part of the data, separate from the training set but taken from the training set, that helps set the weights or other parameters of the model;
  - Cross-validation: use rotating training/validation splits. 1 fold is used as for validation and the others are used for training and this switches until all folds have been used for validation. The final accuracy is averaged over all tests;
  - Baseline accuracy = random guessing. If you have 3 classes it's 33%, if you have 10 classes it's 10%, etc;

    * Accuracy: often might not be a suitable measure - when the classes are very unbalanced (high accuracy in one class might mean low accuracy in the other);
    * Precision and recall per class and average over classes – macro average; per document and average over documents – micro average;
    * F1 score: summarizes precision and recall in one number, it's a harmonic mean. It tends strongly to the smaller of the two numbers, so then both must be high for F1 to be high;
    * Area under the curve (AUC): frequently used in the case of unbalanced classes. Plot the FP against the TP rate and you calculate the area under the curve. It's the probability that the classifier ranks a random positive example more highly than a random negative example;
    * Confusion matrix: a way to examine the classifier's performance at a per-label level. Each (row, column) shows the fraction of times row was classified as column.
- Examples: sorting news articles into topic classes (politics, weather, etc.), sentences into speech acts (assertion, promise, etc.), images according to their content (dog, human, etc.).

3.4 Clustering

- Discovering structure in terms of relationships between items in an unstructured collection of items (documents);
- The input is a data set and the output is a set of clusters that partitions the data points in the set, such that similar items are assigned to the same cluster and the different items are assigned to different clusters;
- The clustering technique can be applied on words, bigrams, POS tags, syntactic tree features – very general;
- Existing bias: the definition of similarity we use for clustering;
- Does NOT automatically output a cluster label. It's up to the user to assign labels;
- Number of clusters is specified in advance. Chosen using hyperparameter search;
- Soft clustering: generates the probability that an instance belongs to each of a set of clusters. Each instance is assigned a probability distribution across a set of discovered categories. Hard clustering is when each instance can only be in one class;
- Model-based clustering is a type of soft clustering. The goal is to design a probabilistic model that captures the latent structure of data and fits the model to that data to obtain clusters;
- The similarity measure:
    - Needs to be symmetric and normalized on some range (usually [0,1]);
    - Cosine similarity: captures the angle between two document vectors plotted in their high-dimensional space;
    - Jaccard similarity: a set metric only capturing the presence/absence of terms with no regards for the magnitude.
- Hierarchical clustering:
    - This is similarity-based clustering: it needs a similarity function to work. Any two objects have the potential to be similar depending on how they are viewed;

  - Build a tree-based hierarchical taxonomy from a set of unlabeled examples (dendogram). Taxonomy: system for naming and organizing things (dimensions are based on observations);
  - Each document can only belong to one cluster (hard clustering);
  - A recursive application of a standard clustering algorithm can produce a hierarchical clustering;
  - Agglomerative hierarchical clustering:
    * Start with each example in its own cluster and iteratively combine them into larger clusters (bottom-up). While there is more than one cluster, find the two most similar ones and merge them;
    * Single-link similarity: merges the two clusters with the smallest minimum distance. Results in looser clusters. Sensitive to outliers;
    * Complete-link similarity: merges the two clusters with smallest maximum distance between elements. Results in tight and compact clusters. Sensitive to outliers;
    * Average-link similarity: compromise between the other two. Takes the smallest average distance between two clusters.
  - Divisive hierarchical clustering: start with one large cluster and you divide into smaller and smaller ones (top-down)
- k-means clustering:
  - A top-down divisive clustering algorithm;
  - Form initial clusters, iterate (repeatedly reallocating instances to clusters in order to improve the overall clustering), stop when the clustering converges or after a fixed amount of iterations;
  - Set K centroids and iteratively reassign the documents to each one until the change in cluster assignment is small or nonexistent;
  - Can use cosine similarity: distance is 1-cosine;
  - Clusters are usually represented by the cluster centroids (a special document that represents all other documents in the cluster, usually the average of all of their values);
  - Sensitive to the seeds we use and the results can vary a lot based on that. Sometimes the seeds can result in poor convergence rate, so should use e.g. a heuristic to select good seeds;
  - Challenges: might result in a local minimum, noise and outliers are problematic, doesn't scale well to large datasets.
- Evaluation:
  - Internal measures: related to the inter/intra cluster distance. The sum of distances between objects in the same cluster are minimized and the distance between objects of different clusters are maximized;
  - External measures: related to how representative the clusters are to "true" classes (ground truth). It can be measured in terms of:
    * Purity: the number of items of the majority class;
    * Rand-index: compares pairs of points in the ground truth to pairs of points in the predicted set;
    * Entropy and f-measure.

3.5 Topic Modeling

- Discovering the main themes in an unstructured collection of items (documents). The themes aren't predefined – they are discovered;
- Primarily used for text;
- Topic models can organize the collection according to the discovered themes, which can be just sub-structures that don't actually give the impression of being a classic "topic" (so the "themes" aren't guaranteed to be meaningful);
- Topic models are part of a larger field of probabilitistic modeling. In generative probabilistic modeling, the data is treated as arising from a generative process that includes hidden variables. Each topic is defined as a probability of words (a vocabulary set, probability for each word of how likely it is for it to occur in this theme);
- Besides the text, we can make use of other non-text data as additional context for analyzing the topics. Metadata such as time, author, location, etc. can be used to analyze topic patterns;
- Generally 2 tasks: discovering the number of topics from the collection, and figuring out which documents cover which topics to what extent. So, the number of topics is pre-defined;
- Topics as terms:
  - Simplest way is to define topics as terms (words or phrases). Then we analyze the coverage of the topics in each document based on the occurrences of these topical terms;
  - Parse the data in the collection and make every word be a candidate topic. Using a scoring function we'll quantify how good each term is – we want to favor representative terms, so ones that appear frequently but actually contain information about the content. To do that we can use TF-IDF weighting;
  - We discover the k topical terms by taking the k terms with highest scores (although when we pick the terms we make sure that they are not too similar to the already picked ones);
  - To compute the coverage of each topic in each document, we can just count the occurrences of each topical term;
  - Problems: when we count words belonging to a topic, we want to also consider related words. Words can be ambiguous and have multiple meanings. If we only describe the topics with 1 word, we can't have very complex topics.
- Topics as word distributions:
  - Fixing the problems of before: using more words to describe the topic, introducing weights on words, and splitting ambiguous words to allow them to be used to potentially describe multiple topics;
  - This can be achieved by using a probability distribution over words to denote a topic;
  - Representing a topic by distribution of words can involve many words to describe the topic and model subtle differences in topics;
  - Since the representation is a probability distribution, we use probabilistic models.
- Models:
  - Naive Bayes model:
    - First we generate documents;

    - For each of M documents, we choose a topic, then choose N words by drawing each one independently from a multinomial conditioned on the topic;
    - Limitation: we can only have one topic per document.
  - Probabilistic latent semantic analysis model (plSA):
    - A model in which each topic occurs with a weight. Can have documents with multiple topics;
    - For each word of the document, choose a topic according to a multinomial conditioned on the document and generate the word by drawing from a multinomual conditioned on the topic;
    - Limitation: number of parameters grows linearly with number of documents in the training set. Also, there is no natural way of handling unseen documents.
  - Latent Dirichlet allocation:
    - Discovers the topics in terms of words associated with those topics. You have to inspect the words that the algorithm has grouped together and give them a label manually, the model won't give the topic by itself;
    - LDA sets a prior on the distributions of topics within the document (what we know about how the distribution should be shaped);
    - Simplex: a space such that all components add up to 1. The prior has to exist somewhere within it; it shows areas with higher probability with darker color;
    - Type of a mixed-membership model: each document is modeled as having multiple components in different proportions;
    - Can be computed automatically and doesn't require any manual effort (unsupervised approach);
    - Assumptions made: the order of words doesn't matter, the order of documents doesn't matter, the number of topics is known and fixed.
- Evaluation:
  - Log-likelihood and model perplexity are two common (predictive!) measures. If the model generalizes well to new data, it will have high likelihood and low perplexity;
  - Human judges can assess the coherence of topic-word distributions.

3.6 Ranking

- Items get ordered with respect to a criterion;
- In IR this criterion is the match with the user's information need (the relevance).

3.7 Representations

- The representation isn't the world! We represent the world and talk about its representation not about it;
- Documents are represented by feature vectors. The simplest model is bag-of-words (unordered list of all words in document collection);
- Representation can be simply having an entry of 1 if it appears in a document and 0 if it doesn't. Beyond binary vectors we can use the term-frequency (TF) vectors and inverse document-frequency (IDF) vectors;

- There are many ways to create the components of the vectors; that's because the representation choice is task, data and model dependent;
- Vectors are lists of numbers and the similarity between them can be calculated using the angles between them when drawn as arrows in space;
- Vectors representing documents should be chosen such that the angles between them are small if the documents are related. Orthogonal vectors are at a max distance from each other;
- Creating representations involves choosing the values of the vector components (weights).

3.8 Word Embeddings

- The input is a word and the output is a vector (hence word2vec's name);
- Can use a 1-hot vector (in them, there is only one entry with a 1). 1-hot vectors for individual words in the vocabulary would be orthogonal to each other in the vector space;
- Word embeddings are vectors of numeric values which they represent words in a lower dimensional space;
- Good when we want to group similar words together in vector space and as an input representation for other algorithms;
- Disadvantage: the components of the vectors aren't interpretable;
- Given a collection of text, we must find a vector representation for all words that maximizes the probability of each word predicting the other words in a window around it;
- Word embeddings capture the context in which words are used in a collection of text. They leverage the distributional hypothesis (words that occur together in the same context are related/have similar meanings) and so they capture synonyms, functionally related words, words from the same paradigm;
- Bias in word embeddings (bias = prior information): it can be problematic/harmful if it's prejudiced, stereotyped. To reduce it, we specify the axis of bias and words to debias and transform the word embedding matrix such that these words are perpendicular to the bias axis but all other words retain their relationships (this doesn't work for biases we're not that well aware of, though);
- Another problem with word embeddings is representing things like negation (e.g. unlike).

3.9 Semantic Resources

- Recognizing phrases with the same meaning is crucial for many text mining applications;
- Semantic analysis:
  - Focus on how parts of a sentence/text are related semantically;
  - On multiple levels: words (word sense disambiguation), sentence (semantic parsing, semantic roles), discourse (text structure representation, co-reference, dialogue acts);
  - Word senses:
    - Homonymy = words that sound the same but have different meanings. Polysemy = different words but related meanings. Many such concepts introduce vagueness in natural language;
    - Many words have more than 1 meaning and we need context and lexical information to get the correct sense. A lexical-semantic resource is needed;

    - We can use a dictionary but it's problematic: word definitions are circular (describe words in terms of other words). Also, there are different sets of descriptions in different dictionaries and some meanings of a word are more similar to each other than other and we can't express that in dictionary;
    - Constructive resource like WordNet: all words with the same meaning are grouped in synsets (collection of synonyms).
  - Sentence meaning:
    - A sentence expresses/communicates about an event;
    - Thematic roles: finding the meaning of a noun in an event. Problems: people don't agree on a standard set of general roles, there are many different theories leading to different rules and resources;
    - There are two directions for semantic roles: generalized (the agent role, applying to as many verbs as possible; arguments that have the most of some characteristics are specified as such) and specialized (specific rule for every verb and event).

4 Week 4

4.1 Information Extraction

- IE – automatically extracting structured information from unstructured natural language text. This "knowledge base" can help one of the limitations of AI: having too little knowledge;
- It's goal-oriented and task-specific and has many sub tasks;
- The structured information consists of entities (uniquely specified objects in the world, such as people, places, organizations, times), relations (a predicate and two arguments), events (something happening at a certain time – involve multiple typed arguments);
- Knowledge is information in context, reusable information;
- Context – location in the world, time (past, present, future), other information dots;
- Goal: link information from unstructured text to a knowledge base so that we create structured information;
- Steps:
  - Entity detection: recognize entities mentioned in the text (recognize start and end point of the entity phrase in the text);
  - Entity linking (link the entity to a knowledge base);
  - Knowledge discovery: detecting implicit relations in unlinked literature; something that hasn't been found before.
- Evaluated often by the correctness of the resulting knowledge base, not by how many sentences were correctly parsed: macro-reading. (As opposed to micro-reading which cares about analyzing each sentence correctly.)

4.2 Entity Recognition

- Information representation in named entity recognition (NER):
  - We represent the entities we recognized with the help of a markup language (e.g. Stand off XML);

  - We can use BIO tagging: each token is a word or punctuation mark labelled with BIO encoding: B-X (first word of an entity of type X), I-X (non-initial word of an entity of type X), O (word outside of any entity).

- How to recognize the entities?
  - We can use a list of entities (like a name dictionary) and label them in the text;
  - Problems: can have the same name referring to multiple entities or multiple names for the same one (ambiguity and variability); the list is also limited and listing all variants means the list will be endless;
  - Better to use a feature-based approach, using the context of words by creating a set of manual rules (this only works for restricted domains);
  - Can also train a supervised machine learning algorithm or NN on a set of annotated examples.
- Evaluation: use precision and recall to evaluate the performance of the system;
- Event detection: identifying events (something that happened at a specific point in time and is worthy to be reported. Has a vague definition because it's very task-dependent and has a duration or time aspect. It's often expressed by predicates (action verb) or nouns).

4.3 Entity Linking

- Link the entities in the text to their entry in the knowledge base: e.g. if the knowledge base is Wikipedia, we link them to their wiki page;
- Linking by ranking: learn a function that learns to rank a small set of candidates. Each candidate is represented as a feature vector that expresses similarity between Wikipedia title page and mention, popularity of the candidate entity, NER label, local context words as a bag-of-words;
- Collective entity ranking: we assume that entity mentions within the same document refer to the same target entity. Disambiguation of one entity might be helpful for the others. We try to optimize the sum of pairwise ranking, but it's computationally expensive so we need heuristics for practical computation.

4.4 Relation Extraction

- Detecting and labeling a relation between two entities in the text; we have two entities and one relation which is verbalized in one sentence;
- We can see it as a binary classification problem: for each pair of entities, determine whether they have a relationship or not, and if they do – what is the relationship;
- Pattern-based RE: fixed scenarios (searching for mentions);
- RE as ML classification task: for a pair of mentions in the same sentence, get: the features of the mentions (frequency of mentions, NER label), distance between mention pair, syntactic relation between mention pair, words/POS between/before/after mention pair;
- Can use this as a way of finding new knowledge about entities (to fill your knowledge base), for information fusion (recognizing multiple mentions of the same relation), open IE: can we find any type of relation in this text?
- Complications: modality and negation are part of natural language and they are difficult to deal with:

  - The degree of certainty, reliability, subjectivity, perspective must be taken into account somehow;
  - Methods for handling negation and modality generally include two phases: detecting negated or uncertain events, and identifying the scope of the negation/modal operator;
  - Detecting negations: matching lexical cues while avoiding double negatives;
  - Scope identification is formalized as sequence labeling problem: each word token is labeled as beginning, inside, outside of a cue, focus or scope span.
- Evaluation: again recall and precision or even f-score.

5 Week 5

5.1 Multimedia

- Wide definition: digital content that comprises more than one modality (audio, video, etc.). The modalities are complementary (together they contain complete information);
- Narrow definition: involves multiple interdependent modalities and is produced by humans for consumption by other humans. This excludes satellite pictures, surveillance videos from CCTV cameras, etc.

5.2 Semantic Gap

- Definition: the difference between the information that a machine can extract from the data and the interpretation that a user in a given situation has for the same data;
  - We see multimedia in terms of events and objects and from there we extract knowledge and then build up higher level wisdom/realization/reflection;
  - The machine sees bits and bytes which it translates into characters and then data structures and documents, sound, image, etc;
  - There is a gap between the image/sound/text/etc. representations and the events and objects we see!
- For example, in music: we can recognize genre, performer/composer, underlying musical work, while the computer can only look at the exact characteristics (performance instance);
- Terminology:
  - In computer vision and multimedia, semantic content are the objects, events and scenes that people see when they look at images;
  - Aspects of the image like framing, composition, aesthetic, style aren't considered semantic content although people can recognize them easily (while computers cannot);
  - So for the semantic gap, the "semantic" is anything you could say that you see when looking at the picture, while in semantic content it refers to objects but also collections of concepts that make scenes and events;
  - How we interpret images is often independent of their semantic content (or "orthogonal" to it).
- State of the art: Google search by image looks good! Until you think about specific user information needs: find a specific instance of the object, find other images of this type object in the same setting or with the same pose, etc.

- The semantic gap is defined with respect to a given situation (which is related to the user's information need). Many problem formulation fail to articulate the situation which leads to failure as well;
- Visual variability:
  - Images can depict the same entity/instance (they are of the same particular object (e.g. my cat)) or depict the same class/concept (they are of the same type of object (e.g. cats));
  - An entity can be fixed (only 1 instance, always sitting in the same place so on all pictures will have the same background), an entity can move (so it's not stationary and thus it can be seen in any place, against any background), an entity can change (change color, shape, etc. e.g. by changing clothes);
  - Identical instances: physically identical entities which can either be differentiated by experts or by people familiar with the entities (we need knowledge);
  - Events are triples of (x,t,p): entities, time, place; they can also be seen as instances and classes; if you don't care about x and t it's called a "scene."

5.3 Multimedia Representation

- Representation: vector that captures certain properties of the content and allows us to process it (for retrieval, classification, mining, etc.);
- Representing audio:
  - Static representation packs variation over time into one value. The audio recording is represented as a vector consisting of the means of the four values over the audio signal (loudness, pitch, brightness, bandwidth). However the characteristics of sound signals vary over time...
  - Dynamic representation: break the audio into windows, classify each (e.g. loud, medium, soft) and count how many windows belong to the each class then create a vector out of these counts;
  - More advanced: Hidden Markov models, recurrent NNs (but there is still some windowing).
- Representing images:
  - As matrices of numbers. A gray-scale image has pixels values in the range 0-255 (usually 0-black, 255-white), and for a color image: each pixel is represented by three pixels values, one for each color channel;
  - Simple vectors of pixels values: unique for every image, so bad for classification – you can't compare them and find similarity, the similarity won't be meaningful;
  - Vectors of global feature values:
    - The components describe the image as a whole;
    - We look at the pixel intensity to find global features;
    - Each component in the vector represents one intensity value; the value of each component represents the number of pixels in the image with that value.
  - Vectors of visual word weights:
    - Storing the local features;

        * We break the picture into patches and then cluster similar pixel patches to create "words" – each cluster is one component of the vector;
        * Look at how often each word appears to make the vector which contains one weight for each word.
    - Vectors of NN values: components are the nodes of a layer and the values are the activations. So, we can use neural networks to extract vectors. With each layer what you extract becomes more and more abstract.

6 Week 6

6.1 Stance Detection

- Stance: the position of someone towards a target (an entity/concept/event/idea/opinion/topic). Stance is people's position on a topic or issue;
- Stance classifier help us understand people, how they debate and argument themselves and represent their stances;
- Orthogonal to topic:
    - There are many other orthogonal-to-topic aspects of text, stances being one of them;
    - Sentiment (affect towards a target; subjective), stance (position towards a target; similar to sentiment but doesn't include emotion), expert opinions (which can also express some sentiment, but it's backed up by evidence and arguments – it's objective; authoritative statement), speech acts (promise, threat, warning, apology, etc), causality (a statement of cause and effect), disinformation (intentionally misleading statements);
    - Sentences with the same topics can be different with respect to these aspects, which aren't mutually exclusive.

6.2 Sentiment Analysis

- Field of study that analyzes people's opinions, sentiments, evaluations, etc. about entities such as products, services, organizations, etc. Facilitated lately by the internet where everyone can express themselves easily;
- We are talking about how people are expressing their views of the world; so we learn about that and about the authors themselves;
- Sentiment expressions are subjective language use (person expresses about their internal private state, which you can't verify);
- Typical SA queries: finding the opinion of a person on a topic, finding positive/negative sentiments for a product (when buying), checking how opinions change over time, etc;
- Interested parties in SA:
    - Businesses and organizations: for product and service bench-marking, policy making, etc;
    - Individuals: interested in others' opinions when purchasing products, or finding political parties they should vote for;
    - Search engines: so they can place ads in the content;
    - In brief: it is useful for decision support, understanding human preferences and aggregating opinions.

- Levels of SA:
  - Document level: classify whether a whole opinion document expresses a negative or positive sentiment. This assumes that the whole document expressed opinion on a single entity;
  - Sentence level: classify whether each sentence has a negative/positive/neutral opinion;
  - Entity and aspect level: finer-grained analysis in which we can actually discover what exactly did the people (dis)like. It's based on the idea that an opinion consists of a sentiment and a target.
- Can have regular opinions (expressing a sentiment only on a particular entity/one if its aspects) and comparative opinions (comparing multiple entities based on some of their shared aspects);
- Opinion words: good, wonderful, bad, awful, etc. Instrumental for sentiment analysis – a list of such words is called sentiment lexicon;
- Feature-based opinion mining:
  - Finding opinion quintuple (g, s, h, t) where g is the opinion (sentiment) target, s is the sentiment about the target, h is the opinion holder and t is the time when the opinion was expressed;
  - The sentiment target can be thought of as an entity with features (aspects) and then it's a quantuple;
  - First do some kind of entity detection to find the entities/events/topics. Then extract information from the entities to get their aspects and features;
  - Aspect expressions that are nouns and noun phrases are called explicit aspect expressions. If they aren't nouns or noun phrases – implicit;
  - Both aforementioned tasks are highly domain dependent.
- Sentiment scores:
  - Most basic and common labeling is: positive/negative/neutral (no opinion or don't know or no answer);
  - Can also do objective vs subjective;
  - Can do emotion classes (joy, anger, etc.);
  - Can do ordinal scale for intensities such as 1-5 or 1-10. It's bad to do 1-100 scale because the difference between e.g. 55 and 56 or 60 is difficult to figure out so different people might use the scale differently.
- Challenges:
  - Sentiment words don't always express a sentiment (questions, conditionals, etc);
  - Sentiment words are ambiguous, context and domain dependent;
  - Sarcasm and irony;
  - Objective sentences can also express sentiments without any sentiment words;
  - Negation markers and their scopes.
- SA evaluation: precision, recall, f-score (computed on positive and negative labels only);
- Opinion spamming: people with hidden agendas can post fake opinions to discredit or promote products, services, etc. without disclosing their true intentions.

7 Week 7

7.1 Authorship Attribution

- Definition:
  - You have text of an unknown author and written samples of which you know the author: you compare them and try to figure out which of those people wrote the unknown text;
  - From ML POV it's a single-label text-categorization using textual features (assigning a class to the text, where the classes are the potential authors);
  - Verification: is this text written by author X? – one class classification. The negative class is chaotic (all other authors apart from the true one);
  - Identification: by which author is it written, a,b,c? – this task can be rewritten into a verification task (is it a? is it b? is it c?).
- Closed/open set authorship attribution: is the true author part of the candidate set or not?
- Author profiling: what is the age/gender/personality type/etc of the author? can we find that out from just the text?
- Authors as generators of text:
  - What's their stylistic fingerprint? The parts of the text that are constant by the same author (not the the content, communicative goal and genre which can all differ per author as well);
  - Human stylome hypothesis: authors can be distinguished by measuring specific properties of their writings;
  - The clues are usually in function words (pronouns, determiners, prepositions, conjunctives) – they are indicative of both your writing style and your personality.
- Application areas: literary studies (resolving disputes on authorship, linking anonymous works to people, analyzing collaborative working), plagiarism detection, forensic linguistics (e.g. authorship verification of text messages);
- Special case of text classification:
  - The difference is that we're not interested in the content of the text;
  - In topic-based classification we are interested in the content words because they carry the semantic information;
  - In style-based classification the function words are most important (articles, prepositions, pronouns, etc). These common words are found to be among the best features to discriminate between authors.
- Cross-domain authorship verification is difficult because the styles can be very different: the domain affects the writing style a lot, the audience can be different, there can be limitations in the length, etc.

7.2 Features

- Lexical features:
  - Split the text in sentences, those sentences in tokens and use those tokens as a profile of your author. So we consider the text as a mere sequence of word-tokens;
  - Advantages: don't need to do any preprocessing, can be applied to any language and any corpus, easy to use feature;

  - Can have bag-of-words features where the words themselves are features;
  - Can also have aggregated features:
    * The counts of the words or counts over lexical characteristics are the features;
    * So we get vectors of numbers - sentence length, word length, etc;
    * The vocabulary richness functions are attempts to quantify the diversity of the vocabulary of the text, like e.g. type-token ratio.
  - Can also use frequencies of sequences of words (phrases; n-grams). This can be powerful for large collections, but the disadvantages are that it doesn't scale and the representation is very sparse.
- Character features (typically this works best):
  - Split the text in characters and consider the text as a mere sequence of character-tokens;
  - Can use characters themselves or their counts as features;
  - Can also have aggregated character-level features: how many digits, how many punctuation marks, whitespace counts, etc., very successful group of features for stylistic purposes;
  - Advantages: language and domain independent, no NLP needed, quite useful to quantify the writing style.
- Syntactic features:
  - Employing syntactic information (how do people write). So it needs some deeper linguistic analysis;
  - Authors tend to unconsciously use similar syntactic patterns, so syntactic information is a reliable authorial fingerprint in comparison to lexical information;
  - Disadvantage: requires robust NLP tools, it's a language dependent procedure and produces noisy datasets due to unavoidable errors made by the parser;
  - It has 3 levels:
    * Full syntactic analysis. The use of the found patterns is useful but isn't often used because it needs accurate syntactic parsing;
    * Looking at chunks only (chunking): shallow parsing of sentence into phrases (NP, VP, PP). It can be extracted automatically with relatively high accuracy, The aggregated features are NP counts, VP counts, length of NPs, length of VPs, etc;
    * POS tagging: use a tagger to label the text automatically. The aggregated features are POS tag frequencies, POS n-gram frequencies, etc. Can also add POS-labels to terms in the bag of words.
- Semantic features:
  - Semantic role labeling and patterns of semantic roles. So it needs some deeper linguistic analysis;
  - Gives more topic than style related information which can be used for author profiling.
- Application-specific features:
  - It can only be defined in certain text domains or languages;
  - Used to better represent the nuances of style in a given text domain such as e-mail messages, SMS messages, online-forum messages;

  - Structural features include paragraph length, type of signature, use of indentation, etc.

7.3 Authorship Attribution Methods

- Profile-based:
  - Take all your text examples from author A, extract features and make one profile to represent this author; same with author B , C, ...;
  - A new text you compare to the profile of the authors and you decide which author it's most similar to;
  - Compression method (ZIP method):
    * Take all texts of author A and compress them;
    * Add the new text to the texts of author A and compress them;
    * Measure the difference of the zip files;
    * Do the same for all possible authors and compare the differences. The smallest one is the one that gives the real author;
    * Originally it was developed for language identification.
  - Common n-grams method:
    * Extract all character n-grams from all texts of each author, create an author profile of the most frequent n-grams;
    * The profile size (l) and the character n-gram length (n) should be tuned with best results obtained for $1000 \leq l \leq 5000$ and $3 \leq n \leq 5$
    * If the training data is reasonably balanced, this works well.
- Instance-based:
  - Most commonly used method in authorship attribution;
  - Using the vector space model: each text is considered a vector in vector space and you can compute the cosine distance between them;
  - Using any ML algorithm: suitable for text categorization tasks. The algorithm should be able to handle high-dimension, noisy and sparse data (SVM, NN, etc);
  - If we have 1 long text per author, we can still use an instance-based method – we can just cut the text in pieces. However too short samples lead to lower accuracies (so don't go lower than 500-1000 words).
- Hybrid approaches:
  - Methods that borrow some elements from both of the other approaches;
  - Example:
    * Representing the training text samples separately (as in instance-based approaches);
    * The representation vectors for the texts of each author are feature-wisely averaged and produced a single profile vector for each author as happens with the profile-based approach;
    * The distance of the profile of an unseen text from the profile of each author is calculated by a weighted feature-wise function.

8 Week 8

8.1 Summarization

- The goal is to take one or more long document(s) and output a shorter, concise version of the document(s) capturing the most important parts;
- We need to understand what is present in the text, have an idea of the semantics of the text. We want a semantic compression of the text;
- To write a concise and fluent summary you need to reorganize and merge information in different sentences into one sentence;
- Even for humans this isn't a trivial task, so it's beyond the state of the art for automatic summarization. We can't expect the current systems to do better than humans;
- Useful because in many cases you cant read everything you want (scientific articles, books, etc) and you still want to know about it, or maybe you want to know the general opinion of a product (summarizing multiple documents) etc;
- When summarizing text, we distill the most essential information from a source text with the purpose of producing a shorter version (abstract/extract). There is a reduction of information depending on the degree of compression; this is given by the ratio length of summary / length of source;
- Input factors:
  - Can have single document or multiple ones;
  - Can be in one language or multiple, or even getting a summary of documents in one language into another language.
- Purpose factors, why are we making the summary (depending on needs, different types):
  - Indicative: "about", explains the topic of the text, so that the reader will know whether they want to read the text to find out more;
  - Informative: "informs", captures the most important information of the text. It can replace the original text completely when it comes to content/message;
  - Evaluative: "judges", evaluates the content of the text;
  - General vs user-oriented: the generic summary considers all the information found in the documents, while the user-oriented one is personalized and focuses on certain information from the source document(s) that are consistent with a user query;
  - General purpose vs domain specific, time dependent vs not.
- Output factors:
  - Extract: a summary composed completely of material from the source. It contains snippets from the source;
  - Abstract: summary that contains material not originally in the source, such as paraphrases;
  - Forms: a single paragraph (classic), a set of selected messages, a set of words (or word cloud), bullet points, etc.

8.2 Summarization Methods

- Full interpretation should be avoided;
- Extractive summarization:

  - We select the most important sentences using feature representations: classification (yes/no for whether to select the sentence) or ranking (assign a score to each sentence and select the top-k) task. Need to conduct a redundancy step afterwards - removing the repeating information;
  - Feasible, but limited in terms of fluency and fixes are required after sentence selection;
  - Methods:
    * Sentence clustering:
      · Cluster sentences based on their similarity then select one representative sentence from each cluster;
      · The idea is that clusters with many sentences represent important topic themes in the input. Selecting one sentence per cluster reduces the redundancy;
      · Disadvantage: every sentence is assigned to one cluster while multiple topics may be addressed.
    * Graph-based methods for sentence ranking:
      · Represent the input as a graph: nodes represent sentences and edges express cosine similarity between nodes (thicker means more weight which means more similar);
      · You can still use clustering (using graph clustering techniques) but this is more flexible because sentences can belong to more than one cluster.
    * Supervised learning task:
      · Features: word count, position in the document, punctuation, cue phrases (to sum up, importantly, especially) etc. This is easy to implement and doesn't need a lot of training data;
      · Using CNNs or RNNs as classifiers: the advantage is that there is no feature engineering (we use word embeddings as input features). This requires a lot of training data and computational power.
  - Problems with extractive methods:
    * Can select sentences that contain unresolved references to sentences not included in the summary or not explicitly included in the original text;
    * There are improvements needed after sentence selection like sentence revision, fusion, compression, ordering;
    * The neural-based approach need training data often, and can fabricate information.
- Abstractive summarization:
  - Learn a text-to-text information model. The training data is pairs of longer and shorter texts. We can use RNNs, translation models, learning a mapping between an input sequence and output sequence;
  - It is much more difficult although more feasible with deep learning models now. There are risks of deviating from the original text (they tend to fabricate information);
  - We want to create a language model with a parameter representing the probability of each word occurring, and then to create our own text we draw words from this probability;
  - Using neural networks:

    - There are two parts: encoder (captures the meaning of the whole document), and decoder (generates a shorter version of the text);
    - Cons: need a lot of training data, requires a lot of computational power, the meaning can be altered because of abstraction/rephrasing.
- Baseline system (LEAD): take the first sentences from the documents; a really strong baseline!

8.2.1 Evaluating Summarization Models

- Intrinsic methods:
  - Test the summarizer on its own task:
    - Criteria: coherence and informativity;
    - Should have no redundancy;
    - Methods: compare to reference summaries or ask human judges (they read the whole doc and compare it to the summary; time-intensive).
  - For extractive summarization: we can use precision and recall (and F1) between the human reference summaries and the automatically made ones;
  - For abstractive summarization: we compute the overlap with human reference summaries using ROUGE (counts the number of words that overlap between the human and automatic summaries). ROUGE often has weak correlation with human judgements but human judgments for relevance and fluency are strongly correlated to each other;
  - The human judges approach is called pyramid method: we score summaries based on summarization content units (SCU) which are determined by human annotation. This requires multiple reference summaries;
  - You can also ask the people about relevancy, informativeness, how easy it is to read and understand, etc. Additionally, we can give them two summaries (one automatic one and one human one) and have them decide which is better.
- Many current neural approaches use the same dataset and automatic metrics, and differ from one another only by small margins. This is problematic because:
  - The task is underconstrained (there is more than one way to write a summary);
  - News articles (used in the training data) have a very specific layout which means the model might learn this bias
  - The data is noisy and can include articles that advertise other articles.
- Extrinsic:
  - Test the effectiveness of the summary within another task by giving the users a task to complete where they need to use a text or its summary;
  - Ask experts about the utility (while doing something else, use the summary to find specific information to see if this is enough, without the original document);
  - Speed gain in a human task.