Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)
Факультет управления и прикладной математики
Вычислительный центр им. А. А. Дородницына РАН
Кафедра «Интеллектуальные системы»

Воронов Сергей Олегович

Вопросно-ответная система с автогенерацией пространственных структур

010990 — Интеллектуальный анализ данных

Выпускная квалификационная работа магистра

Научный руководитель:
д. ф.-м. н. Воронцов Константин Вячеславович
к. ф.-м. н. Кац Борис Гершеневич

Москва

2016

Содержание

Аннотация

Моделирование человеческого подхода к решению той или иной задачи часто используется для построения автоматических алгоритмов. В работе предлагается алгоритм, способный отвечать на вопросы, связанные с пространственным расположением объектов, действия которых описаны в исходном тексте. Новизна алгоритма заключается в том, что для определения ответа производится генерация последовательности кадров о расположении объектов в пространстве и времени, аналогично тому, как это представляет себе человек.

Каждое слово представляется в виде скрытой Марковской цепи. Каждое состояние каждой модели имеет заданное распределение значений признаков (признаки зависят от расположения объектов на кадре), с помощью которых рассчитывается правдоподобие пары (кадр видео, состояние модели слова).

Для генерации последовательности кадров, наиболее точно описывающих полученный набор моделей слов, используется генератор случайных траекторий объектов, основанный на алгоритме Метрополиса-Гастингса. Функцией качества полученной последовательности кадров является произведение правдоподобий моделей слов на наиболее вероятном распределении их состояний по кадрам.

Построенная система в состоянии отвечать на набор вопросов о изменяющихся во времени пространственной структуре объектов, изначально описанных в текстовой форме.

Аннотация

To build algorithms for automatic problems solving, one often models human notions about problem solving procedure.

In this paper we construct an algorithm that answers questions about the spatial structure of objects, actions of which are described in the input text. The novelty of the algorithm lies in generation of additional visual information, just as a person responds to this type of questions, imagining spatial scenery in their mind. A sequence of frames is generated according to the similarity to the sentences of the source text. We use the three step answer generation procedure. The first step is the generation of all possible answers. The second step is the matching with the obtained visual information. The third step is to select the answer, which has the maximum answer-video likelihood.

Every word from the original text is represented as a linear hidden Markov model. For each state of the each word model there is a prior distribution of feature values (features depends on objects on a frame), which is responsible for the likelihood for (video frame, state of word model) pair.

To generate visual information that would be the most similar to the obtained set of models, the random generator of objects trajectories, based on Metropolis-Hastings algorithm, is used. The product of likelihoods of the most probable path through word model states is used as the quality function of obtained video.

The constructed system can answer a set of questions about the time-varying spatial structure of objects, described in text form.

1. Introduction

How does a typical Question Answering system work? Firstly, it performs question analysis. Secondly, the system produces a query for its database (it could be stored outside the whole system, like the Internet). And the last step is ranking answers (to return the best one). Today, many people use AI assistants (like Siri or Google Now) in their daily routine. But these systems can only extract information from the knowledge base, but they can not imagine your movements from natural language and speech. In this paper we propose an approach that helps the machine to understand movements of object from text (these movements would be generated) and to answer questions about that.

According to [1], typical QA systems can be divided into three main categories:

Linguistic approach. The first QA systems (1960s) were NL query front-ends for knowledge database (like BASEBALL [2]). Database size imposed a limitation: these systems were able to answer questions only inside a restricted area. Next step of evolution was acquiring the Internet as a knowledge database (examples: START [3, 4], [5] and [6]).

Statistical approach. Size of data, created by mankind, has increased importance of statistical techniques. They have been applied to different stages of a QA system (analysis of question type, predictions about expected answers, etc). Famous systems are IBM's statistical QA [7], [8]. According to [1], [9] has investigated the prospects of applying statistical methods to answer finding task in QA and discovered that these techniques performed quite well depending on the underlying data set characteristics – vocabulary size, the overlap between question and answers, between multiple answers, etc.

Pattern matching approach. Here one could use text patterns to answer some types of questions. For example, the question «Where is Disney located?» relies on pattern «Where is <object> located?» and produce answer «<Object> located at <Location> ». Many of the QA systems automatically recognize such text patterns from text passages rather than employing complicated linguistic tools to text for retrieving answers.

Idea of extending text QA to work with some types of media is pretty natural: there is a lot of information in the Internet in media form. In [10] authors try to enrich text answer

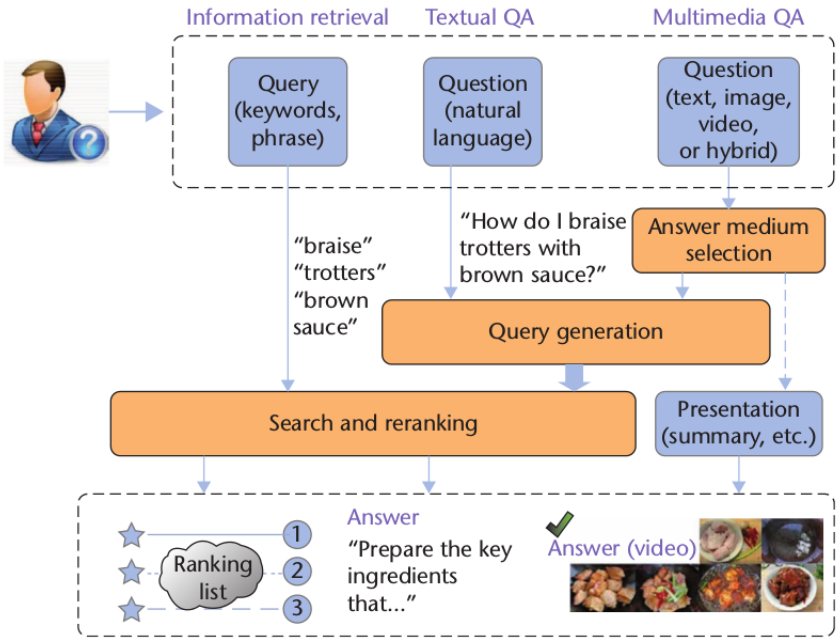


Рис. 1: Difference between Multimedia QA and other QAs (image taken from [13])

with media (image or video) information from internet. The paper [11] tries to work only with key shots from video. The study [12] used NLP to create similar questions and rerank obtained videos queues with video analysis.

Some papers provided idea of answering questions about images or videos. For example, in [14] authors built Spot – a system able to answer specific questions about surveliance videos. START translated an English question into inner representation (video filter) that used to transform (dynamically) raw video [3]. Also, neural architectures were used for QA about images [15].

This paper solves the problem of answering questions about movements of objects by generating video-like info, describing these movements. Thus, we provide an approach for QA that is similar to human thinking: generation of a video-like scene while answering the question. We use the idea proposed in [16] about word representation as multistate FSM (hidden Markov model) over features extracted from video (see Section 3). Same as [17], we represent all words as HMM, not only verbs. Using START [3] we construct sentence-specific structure from HMMs, representing separate words, considering the relations between the objects. Following [17], in Section 4 we create a function $S : (B, s, \lambda) \to (\tau, J)$, where B

represents the information extracted from a video clip, s represents the sentence, λ represents word meanings, τ is the video-sentence score, and J is a collection of tracks, one for each participant in the event described by the sentence, corresponding to the best video-sentence score. In Section 5 we describe track generation procedure for given sentence. In Section 6 we use this generation procedure to answer hypothetical spatial questions.

2. Definitions

2.1. Frame and Video

Let frame be RGB-image with rectangular detections of objects. Each detection is assigned to some object class, like chair or person. Video is a sequence of T frames.

2.2. Features

A feature is a function that maps objects positions into finite set of values. In this paper two type of features are used:

1. features that depend on single object (e.g. velocity orientation)
2. features that depend on two objects (e.g. distance between objects)

Consider feature **distance between objects**. There is a real distance between objects and the distance in the image is a proxy for that real distance, so we quantize the image distance into 5 bins which one can interpret as:

- very close
- close
- medium
- far
- very far

For every pair of objects on frame fr we compute distance between them and further assign one of the feature **distance between objects** values.

We extract these features from objects tracks and use them in Word Models.

In this paper we used next features:

1. class
2. distance between objects

3. relative orientation

4. relative position

5. relative velocity

2.3. Viterbi algorithm

The Viterbi algorithm [18] is used when one wants to calculate the most probable path through the states of HMM over time. Main problem here is that with n states and T moments of time we have about n^T (at least by grown rate) probable paths. If one tries to calculate probabilities through all of them he faces lack of computing power. Viterbi's idea is that, if we think about state k in moment t, we can consider only one way of max probability (and throw others). Therefore, if several paths converge at k in moment t, one do not have to calculate every path, and all, except the most probable way, could be discarded from future computations.

In fact, instead of computing all length T paths from k^0 to k^{n-1} we will compute values $v(i,t)$ – the highest path probability reach k^i from k^0 by t iterations. It means

$$v(i,t+1) = \max_s \left[v(s,t) \cdot p(s,i)\right], \tag{1}$$

where $p(s,i)$ is probability of moving from k^s to k^i. Number of calculations will be $O(TS^2)$, where S is max output degree among states. This is much better than S^T from baseline method.

One can see here there are no restrictions for function p. Also, we could use any scores for states. This changes Equation 1 into

$$v(i,t+1) = score(i) \cdot \max_s \left[v(s,t) \cdot p(s,i)\right]. \tag{2}$$

Lets solve the following task: find the highest probability of length T path from the 0th state to any state. This algorithm could be described as:

```
v(i, 0) = 0;
v(0, 0) = score(0);
for t = 1 to T do
    for i = 1 to n do
        v(i, t) = 0;
        for s = 1 to n do
            if v(i, t) < v(s, t − 1) · p(s, i) then
                v(i, t) = v(s, t − 1) · p(s, i)
            end
        end
        v(i, t) = v(i, t) · score(i);
    end
end
```

Algorithm 1: Viterbi algorithm for HMM with transition function p and *score* of states.

The Algorithm 1 could be used to solve Equation 2 after logarithm transformation. This transformation allows us to solve the following problem: assume we have a score for every object detection on frame and transition score between these detections on adjacent frames. We want to find one track though time among these detection. Track is a sequence of detections (one per frame) with small sum of distances between object detection (described in 4.1.1) on adjacent frames. Therefore Viterbi algorithm could find the best track.

3. Word model

We want to describe time-varying series of actions (i.e. obj_1 **picks up** obj_2). Assume that this action is represented as movements obj_2 near obj_1 on frames of video. We want to introduce single model for each word which consists of list of states, and every state will linked with set of objects positions (usually it's their position on frames). Every state of model describes part of action. A linear HMM [16, 19] will be used here. Note that the number of dependent objects is constant among states of the model (because every part of action should involve the same objects as previous). One can see Word Model scheme on Figure 2.

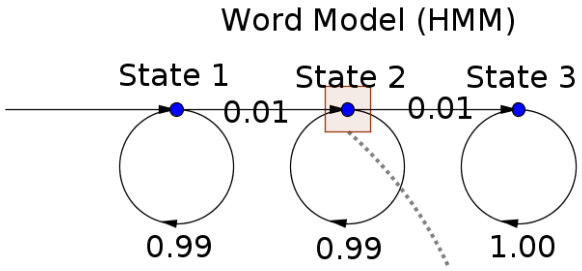


Рис. 2: Word model

Consider two examples of **Word model** for word **pick up** and **approach**. Every state of first model is linked with part of whole **pick up** action:

$state_1$: obj_2 is much lower than obj_1 on the frame

$state_2$: obj_2 is lifts up (to obj_1)

$state_3$: obj_2 is close to obj_1

Note that during these actions obj_1 should be more or less stationary.

Similarly, **Word Model approach** states will relate to

$state_1$: obj_2 is far away from obj_1 on the frame

$state_2$: obj_2 is on medium distance from obj_1

$state_3$: obj_2 is close to obj_1

Note that during these actions obj_1 do not have to be stationary.

Usually, every simple action could be split into 3-4 parts. It means, by default we have much more frames than part of action. Therefore, every action (and, accordingly, every model state) will be linked with continuous sequence of frames. To find how well state of **Word**

Таблица 1: Feature **distance between objects** distributions inside states of **Word Model** approach

feature value	p in st_1	p in st_2	p in st_3
very close	0.0	0.0	0.6
close	0.0	0.25	0.4
medium	0.0	0.5	0.0
far	0.4	0.25	0.0
very far	0.6	0.0	0.0

Model fits objects positions on frame we use a set of features, that could measure similarity between state and objects positions.

3.1. Word Model state

How to understand that one state of Word Model fits current frame well? To solve this problem for every feature f we need to determine how well feature value v fits this state. It can be recorded as feature values distribution, e.g. the first state of **approach** Word Model means that objects have some space between. One can see distributions for feature **distance between objects** in Table 1.

We can describe Word Model in the following way:

- number of objects, which positions will be used for computing likelihood
- list of states of HMM (they usually refer to part of whole word action)

Also, every state of model contains the following information:

- subset of features with prior probability distributions on these features values
- probability of moving to the next state and probability of staying in this state (usually 0.01 and 0.99). We do not have to store any other probabilities, because they are 0 (we use linear HMM)

Note, that these features may be applied not for all dependent objects, but for its subset. Example: Word Model **pick up**. In this action obj_1 should not move, so here feature **velocity**

magnitude is used. This feature takes one object position as input, whereas whole model depends on two objects.

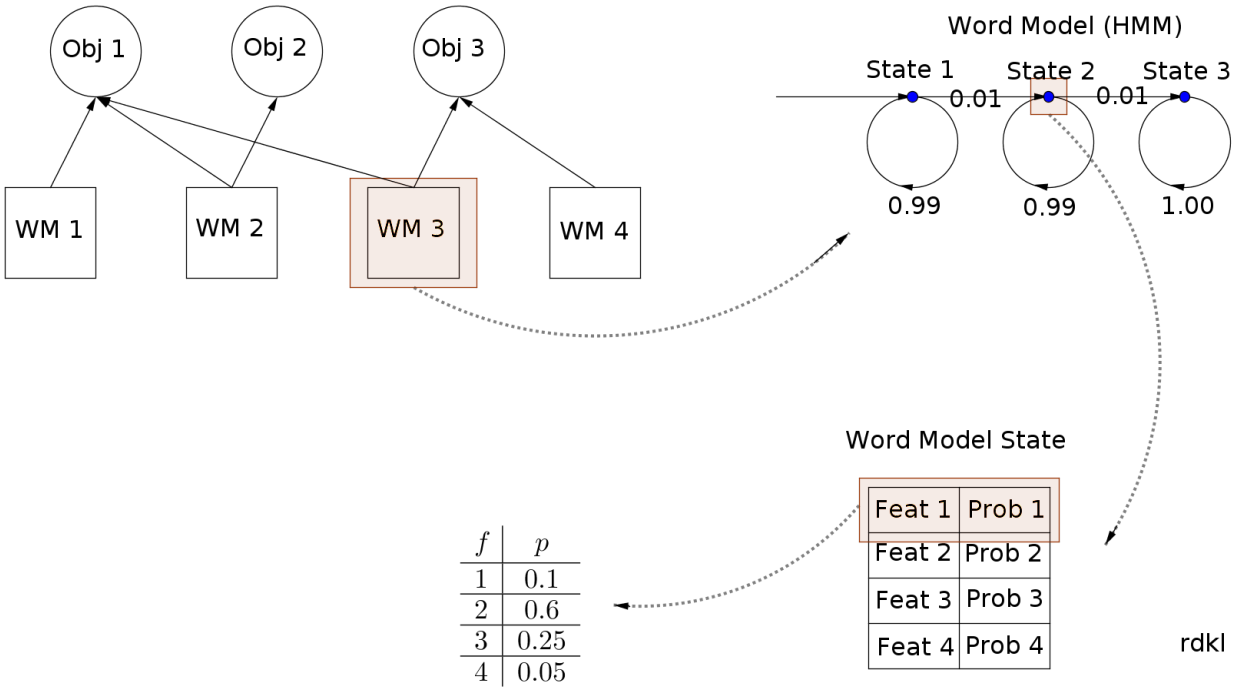


Рис. 3: System overview: connection among Word Models and Objects, Word Model, Word Model state and one feature values distribution over one state.

Left-top. Every Word Model has one or more dependent objects. These connections are shown here as arrow at .

Right-top. Every Word Model is HMM, so it has transition probabilities (that are shown as arrows at right-top).

Right-bottom. State of Word Model: features and its values distributions.

Left-bottom. One feature and its values probability in current state.

3.2. State-frame likelihood

Consider Word Model m, st – all states of model m and objects $objs = [obj_1, ..., obj_d]$, where d is equal to number of input object for model m. Now we are able to compute likelihood of pair $(st, objs)$:

$$d = 1: \quad L_{st-fr}(st, objs) = \prod_{f_1 \in st} f_1(obj_1)$$
$$d = 2: \quad L_{st-fr}(st, objs) = \prod_{f \in st} f(obj_1, obj_2) \prod_{f_1 \in st} f_1(obj_1) \prod_{f_2 \in st} f_2(obj_2)$$

where f is feature, that takes two objects as args, f_1 – feature, that takes first model object as args, f_2 – second. The algorithm do not limit d, but in practice now we use models with $d = 1$ and $d = 2$ only.

3.3. Model-video likelihood

We're able to determine, how good current model fits set of frames. Let q_m be a number of dependent objects for Word Model m with s_m states, and n be a number of frames $fr_1, ..., fr_n$ (with q ordered objects positions on each frame). We want to link every frame with some state of m. If one names states as parts of whole model action, frames assigned to state will show action of this state (of course it will work in case model fits video well).

Assume we have translation the function

$$tr : \{1, ..., n\} \to \{1, ..., s_m\}, \text{ such that } x > y \Rightarrow tr(x) \geq tr(y).$$

The function tr maps number of frame into assigned model state. Then the following Equation holds:

$$L(m, \{fr\}, tr) = \prod_{t=1}^{n} L_{st-fr}(st_{tr(t)}, fr_t) \prod_{t=2}^{n} hmm(tr(t-1), tr(t)) \tag{3}$$

where L_{st-fr} is likelihood of (state-frame) pair, defined in 3.3 and $hmm(tr(t-1), tr(t))$ is probability of moving from state $tr(t-1)$ to state $tr(t)$ for model m.

Equation 3 is used below as evaluation function, describing how good model fits pair (list of frames, translation function).

Due to machine precision, below we will maximize only $\log(L)$. It could be reformulated in following way:

$$\log(L) = \sum_{t=1}^{n} h(k^t, fr_t) + \sum_{t=1}^{n} a(k^{t-1}, k^t), \tag{4}$$

where k^t means state $tr(t)$, $h(k^t, fr_t) = \log L_{st-fr}(k^t, fr_t)$, and

$$a(k^{t-1}, k^t) = a(tr(t-1), tr(t)) = \log hmm(tr(t-1), tr(t)).$$

4. Scoring

In this section we solve the following problem: find how well given sentence S describes given video (list of frames) $V = (v_1, ..., v_n)$. We should solve the following tasks:

- find detections of objects
- transform detections into logical connected tracks
- parse input sentence and produce word models for all words
- find translation function for every model (tr, that was used in Equation 3)

4.1. Object and Tracks Detection

Assume we have a video (or, at least, list of frames). The first problem we faced is to find all detections of all objects on each frame. The idea of joint track and action recognition is taken from [17]. Our task is to find tracks of specific objects with specific properties. Building acceptable image detector is a hard problem and it is beyond this paper scope, so we will use [20, 21] and will try to maximize quality of output detections of objects. This detector works as the black box, that by pair (frame (picture), objects_class) produces list of pairs (object_detection, detection_score). Our approach assumes that we could find at least one acceptable object for each track on each frame. We could easily throw out unused detections. So we will use overgeneration (and will take a big amount of objects from detector).

This subsection task:

input: few classes of objects (e.g. bicycle or car), image detector

output: smooth trajectories (at least one for every class)

We use the same notation as [17]:

- j – one object track
- j^t – object detection assigned for track j on frame t
- $b^t_{j^t_l}$ – detection chosen for track (trajectory) l on frame t.

- $f(b)$ – detection score by image detector
- $g(b, b')$ – score function shows how detections b, b', chosen for one track, fit this track (temporal-coherence score)

4.1.1. One Track Identification

The simplest case: we have detections of one class. We want to choose one detection per frame for smooth object track j. What does «smooth object track» mean? This means:

1. track: list of detection, one per frame
2. object: each of these detections should has acceptable score
3. smooth: distance between object detections on adjacent frames should be relatively small

How could we satisfy «smooth» requirement? One could estimate movements of detection by adjacent frames, because in the common case object movements per frame are relatively small comparing to detection size (usually detection is a box). So, «smooth» means that next object detection position should be near some «estimated» position.

Main idea of using optical flow [22] as a distance correction is to decrease g as much as possible for b, b' from the same track. If b, b' is in one track then optical flow vector direction will be similar to direction of vector $\vec{bb'}$. We used OpenCV [23] Python implementation, that computes a dense optical flow using the Gunnar Farneback's algorithm [24].

So, we will find best track for video B using the following equation:

$$\max_j \left(\sum_{t=1}^{T} f(b^t_{j^t}) + \sum_{t=2}^{T} g(b^{t-1}_{j^{t-1}}, b^{t-1}_{j^t}) \right) \tag{5}$$

This equation could easily be solved by Viterbi [18] algorithm (in fact, dynamic programming). That will produce a lattice size $J \times T$, where J is maximum between number detections on each frame.

In this paper

$$f = \min(1, \max(-1, detection\_score)), \text{ and}$$

$$g = -distance(b, b') + optical\_flow(b, b').$$

4.1.2. Many Tracks Identification

Lets generalize our problem: now we have L different classes of objects (and j_i^t object detections of class i on frame t), and we want to find L different smooth tracks on video B. Then equation for this task is:

$$\max_J \sum_{l=1}^{L} \left(\sum_{t=1}^{T} f(b_{j^t}^t) + \sum_{t=2}^{T} g(b_{j_l^{t-1}}^{t-1}, b_{j_l^t}^{t-1}) \right) \tag{6}$$

This will produce lattice of size $LJ \times T$, that would be solved in polynomial time by Viterbi algorithm.

4.2. Joint Action and Track Detection

We know how to compute model-video likelihood (by video here we mean ordered list of detections with length equal to number of dependent model's objects) from Equation 4. But from video we could only get a list of detections for every class (that should be assigned to models). Also one model represents only one word, but from the sentence we will obtain much more models. Moreover, sometimes we need to change track configuration (if many detection of one objects are generated) using informations from objects. From paper [17] we could get an idea of computing objects, word models states and track assignments together.

Firstly, we could just summarize Equations 4 and 6 (they are still independent) and maximize result to produce best model states $\mathbf{k}$ for found track(s) on frame fr_i:

$$\max_{\mathbf{J}} \left[\sum_{l=1}^{L} \left(\sum_{t=1}^{T} f(b_{j^l}^t) + \sum_{t=2}^{T} g(b_{j_l^{t-1}}^{t-1}, b_{j_l^t}^t) \right) \right] + \max_{\mathbf{k}} \left[\sum_{t=1}^{n} h(k^t, fr_t) + \sum_{t=1}^{n} a(k^{t-1}, k^t) \right] \tag{7}$$

Here L is number of tracks we need for model m, k – states of model m.

Secondly, we need to configure tracks (assume we have two objects tracks here) for model m. To understand why we need this, one could imagine two cars on video. The first car (c_1) is recognized better, than second (c_2), and the first track has bigger score. But this track fits model badly (c_1 just moved on road, c_2 crashed into tree (that is why last frames detections have fewer score), m is **crash**). So we have to choose track for c_2 despite the fact it has lower score. So we will optimize them altogether:

$$\max_{\mathbf{J},\ \mathbf{k}} \left[\sum_{l=1}^{L} \left(\sum_{t=1}^{T} f(b_{j^l}^t) + \sum_{t=2}^{T} g(b_{j_l^{t-1}}^{t-1}, b_{j_l^t}^t) \right) + \sum_{t=1}^{n} h(k^t, fr_t) + \sum_{t=1}^{n} a(k^{t-1}, k^t) \right] \tag{8}$$

Parts of Equation 8 are linked because of $h(k^t, fr_t)$. fr_t here means ordered list of detections from tracks on frame t from video ($[b^t_{j^1}, ..., b^t_{j^L}]$). This also might be solved by Viterbi algorithm: we will produce lattice of size $L \times T \times P^C \times |\mathbf{k}|$, where P^C means product of average number of detections per object class power number of classes (C).

Equation 8 allows us to choose detections-tracks, tracks-models and states-frames synchronized. That leaves only one problem unsolved: we should use more models. But what links Word Model with outside parts of equation? It is translation function that assigns states to frames. What if we build many Word Models into one hyper Word Model, that takes more objects? That means that:

1. $k^t \to (k^t_1, ..., k^t_M) : (\mathbf{k} \to \mathbf{K})$
2. Equation 3 translates into $\prod_{m=1}^{M}$ (Equation 3)

So, we need to rewrite our Equation 8 into:

$$\max_{\mathbf{J},\ \mathbf{K}} \left[\sum_{l=1}^{L} \left(\sum_{t=1}^{T} f(b^t_{j^l}) + \sum_{t=2}^{T} g(b^{t-1}_{j^{t-1}_l}, b^{t-1}_{j^t_l}) \right) + \sum_{m=1}^{M} \left(\sum_{t=1}^{n} h(k^t_m, fr_t) + \sum_{t=1}^{n} a(k^{t-1}_m, k^t_m) \right) \right] \quad (9)$$

It will produce lattice from Figure 4. This Equation is the most general case of this equations type. To solve it with Viterbi algorithm we will produce lattice from Figure 4 (image from [17]).

Recall that h function means how well detections from frame fit Word model and a means translation penalty between states for Word model.

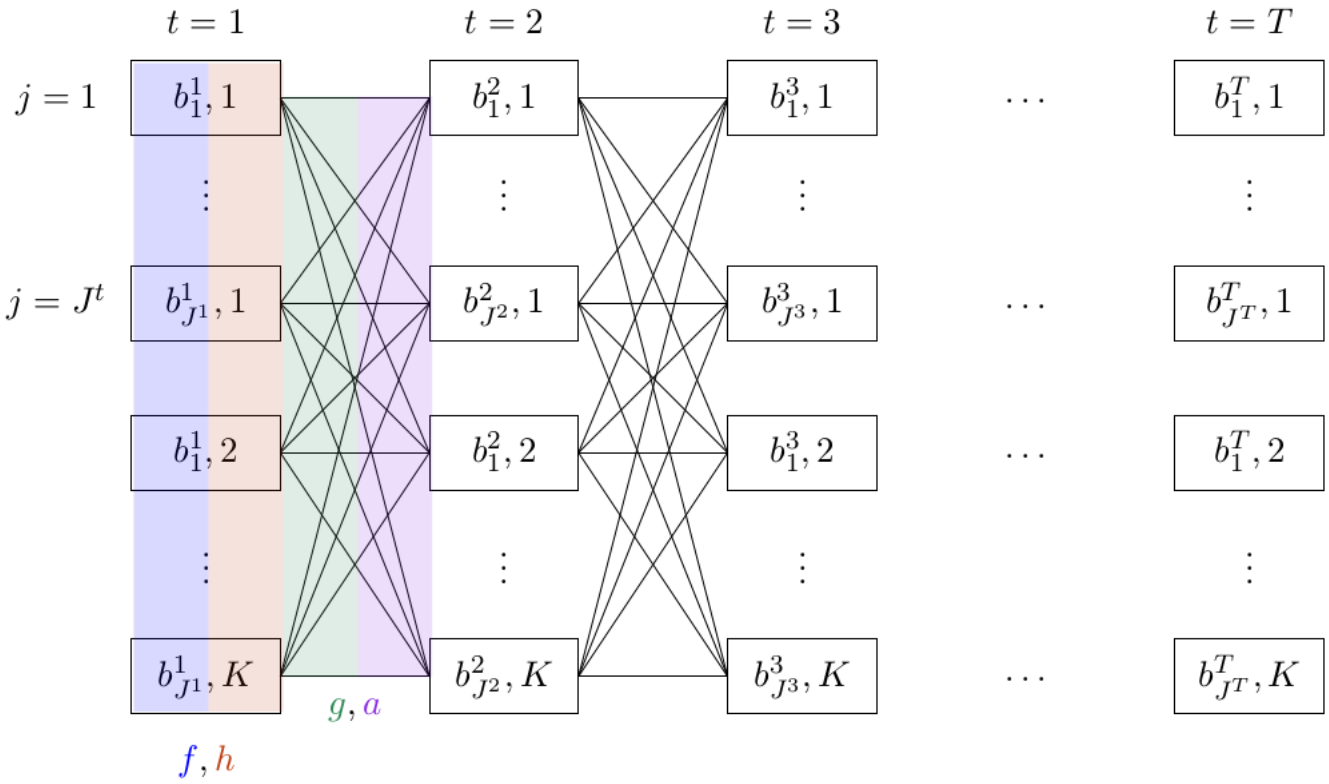


Рис. 4: Final lattice for solving Equation 9. $b_{J^i}^t$ here means object detection, $1, ..., K$ – states of hyper Word model (union).

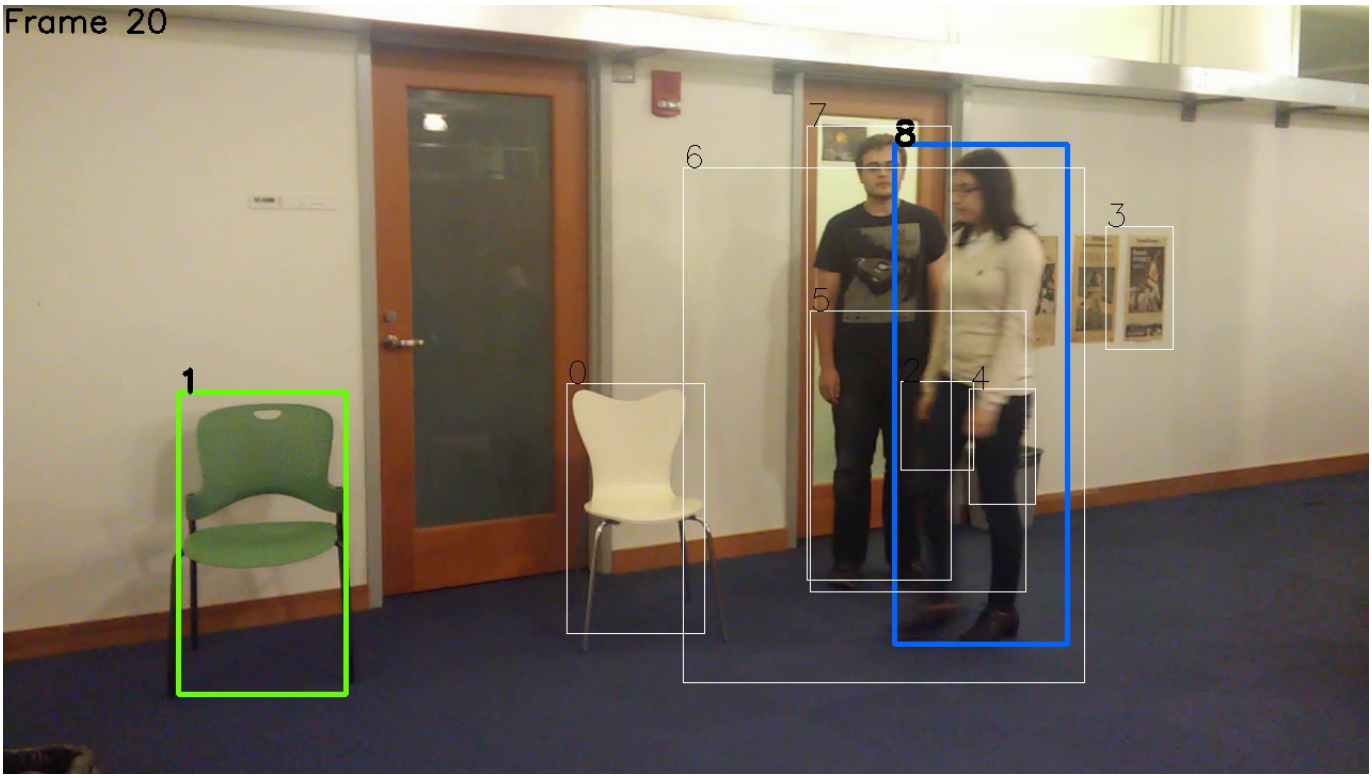


Рис. 5: Joint sentence-video recognition. Detection of sentence «A person approached a stationary chair.» on video with two persons (one person is stationary) and two chairs. Rectangular boxes show detection found on video by [20]. Their count (more than 4) shows overgeneration of object detections, showed in 4.1.

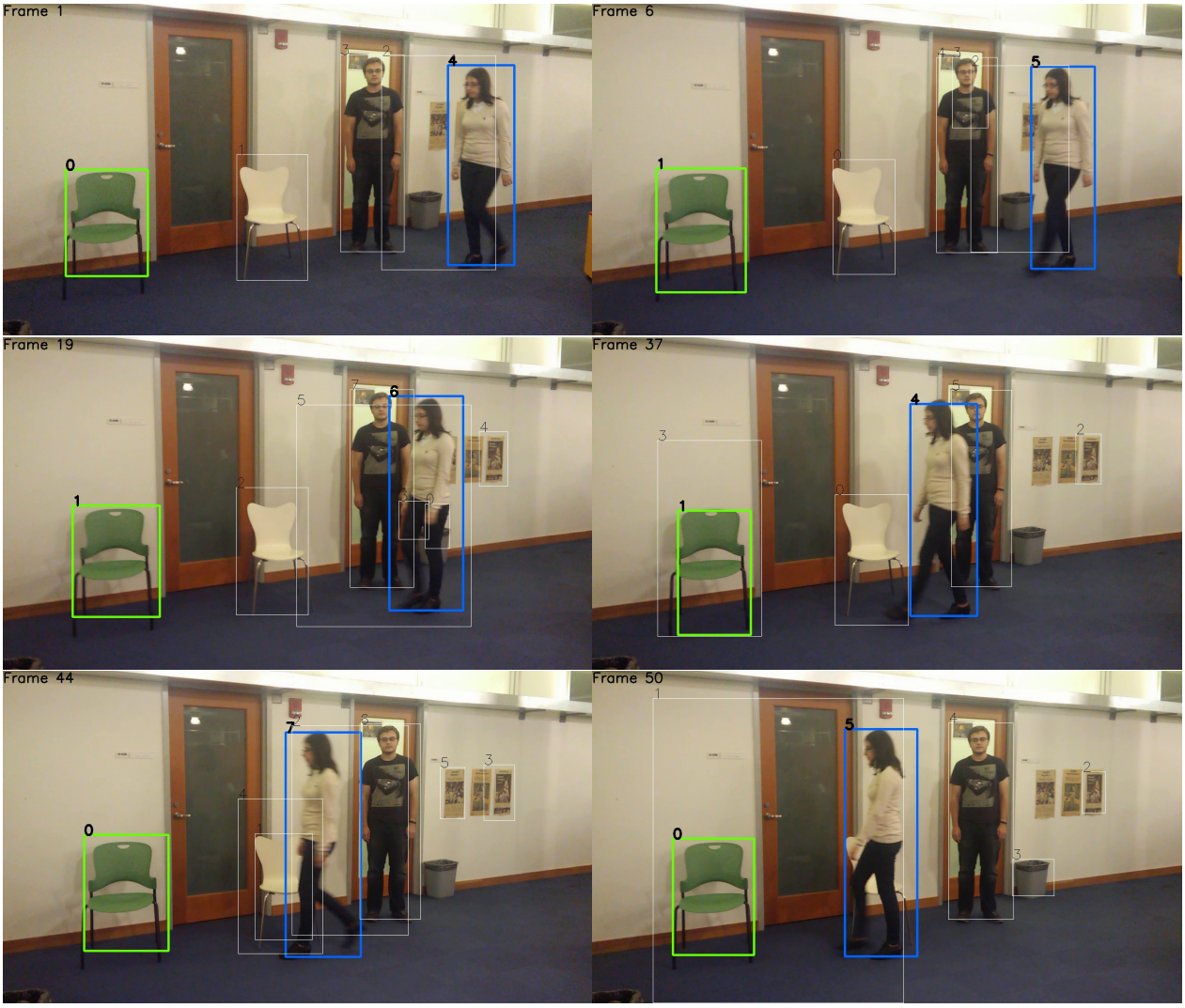


Рис. 6: Joint sentence-video recognition. Detection of sentence «A person approached a stationary chair.» on video with two persons (one person is stationary) and two chairs. Few frames from video are used to show that algorithm is able to recognize events on video.

5. Generation

Our next task is to generate visual information (that is similar to simple generated videos). This information should be similar to the way people think about these questions. We will start from the case, where we have only one model.

5.1. Object Trajectories

We want to generate visual information that would be similar to the simplest videos. That means here will be objects and their movements. But in real videos (with small amount of exceptions) object will move slowly through frames (one might think about video as sequence of frames). Therefore, if we interpolate trajectory of any object through space and time it will be mostly smooth. Lets try to generate something, that would be similar to real trajectories. We will use process of random search of available trajectories space. Due to huge dimensionality of this space there is a necessity of limitation. It produces the following requirement: it would be good if whole trajectory could be controlled by relatively small amount of parameters. That is why B-splines as trajectories were used.

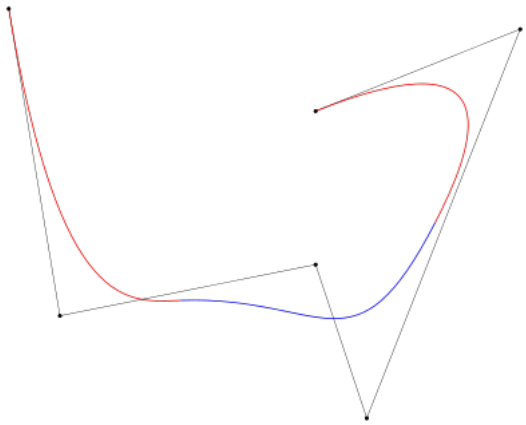


Рис. 7: Example of B-spline. Black points represent control points.

$$B_{i,1}(t) := \begin{cases} 1 & \text{if} \quad t_i \le t < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

$$B_{i,k}(t) := \frac{x - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(t) + \frac{t_{i+k} - x}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(t). \tag{11}$$

Here t means time (or, in our notation number of frame). These basis functions are one-dimentional. To produce 2d picture we use function

$$\vec{C}(t) = \sum_{i=0}^{s} \vec{P}_i B_{i,k}(t) \tag{12}$$

Here $\vec{P}_1, ..., \vec{P}_s$ – is a set of 2d control points. In this paper we took s near 7. To produce

object positions on frame №t we will compute value $\vec{C}(t)$. It will be 2d point means position of object's center. Knots $(t_0, ...t_{last})$ are taken uniform from $[0, T] : t_0 = 0, ..., t_{last} = T$.

5.1.1. New Trajectory Generation

Assume we have function $\vec{C}(t)$ and want to move to the another trajectory. One can understand that if we change control points P_i it changes whole spline, whereas basis functions $B_{i,k}$ remains unchanged. Assume we have s control points, then whole generation procedure:

$2x_{size}$ and $2y_{box}$ are sizes of object box;

for *i = 0 to s* **do**

$P_{new}[0] = U[x_{size}, frame\_size_x - x_{size}]$;

$P_{new}[1] = U[y_{size}, frame\_size_y - y_{size}]$;

$\vec{C}_{new} = \vec{C}$ with replace $P_i \leftarrow P_{new}$;

if C_{new} *is acceptable* **then**

replace P_i with P_{new}

else

next iteration;

end

end

Algorithm 2: Generation of new trajectory

Also, another option of resampling control point was used: normal distribution with center in current position. But this did not increase quality of producing trajectories.

5.2. Tracks Generation

This subsection is about solving the following problem. We have Word Model m, and we have to generate the sequence of frames with the biggest likelihood with m (using Equation **??**). Lets assume m depends on 2 objects (like **approach**). First we will represent every object as rectangular box, that could be described as center position plus size of box.

So, whole algorithm will consist of next two steps (that were repeated while iteration limit is not reached):

- generating trajectories for objects
- checking if trajectories are acceptable

Generation is described in 5.1.1. Lets talk about acceptance process. in this paper two algorithms were used: Metropolis-Hastings algorithm and Simulated Annealing Search.

5.2.1. Metropolis-Hastings Algorithm

As one could understand trajectories space is highly complex, but we want to sample a set of tracks from it. Here we could use the Metropolis–Hastings [25] algorithm (because direct sampling here is difficult).

Generate initial track t;

while *iteration number is not reached* **do**

  Generate track candidate t';

  $\alpha \leftarrow \frac{s(t')}{s(t)}$, where s is track score (in the common case from Equation 9);

  if $\alpha > 1$ **then**

    replace t with t'

  else

    accept the candidate with probability α

  end

end

Algorithm 3: Metropolis-Hastings track(s) generation

But if we want to get more than one tracks we will generate them together, because we do not need good track for object one and good for object two, we want to get set of tracks that fits our sentence together.

5.2.2. Simulated Annealing

First described in [26], main idea of this algorithm used in paper is: use temperature function to increase oscillations of score. This helps algorithm to leave local max. One could see Algorithm 4.

Generate initial track t;

while *iteration number is not reached* **do**

  Generate track candidate t';

  $\alpha \leftarrow exp\left(\frac{s(t') - s(t)}{T}\right)$;

  $T \leftarrow 0.99T$;

  $\alpha \leftarrow \frac{s(t')}{s(t)}$, where s is track score (in the common case from Equation 9);

  if $\alpha > random(0, 1)$ **then**

    replace t with t'

  else

    next iteration;

  end

  if *T is near 0* **then**

    save track;

    resample T;

  else

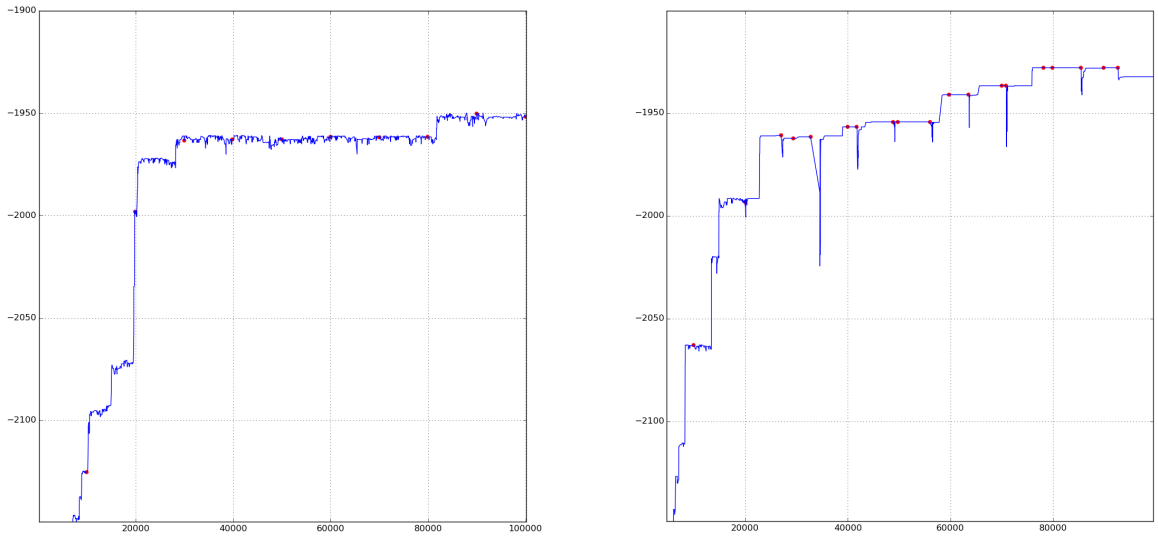
  end

end

Algorithm 4: Simulated annealing track(s) generation

5.2.3. Scoring function

One can see Algorithms 3 and 4 use scoring function for generated tracks (usually from Equation 9). For generation purposes we use $f = 1$ and $g = 0$ (part of Equation 9 produced by Equation 5 will be constant). $f = 1$ means (comparing real video problem) that all objects have maximal detection score. $g = 0$ means that tracks will not interfere amongst themselves (if two position of one object were obtained from the same track then they will be assigned to one track by solving Equation 9).



(a) Metropolis-Hastings (b) Simulated annealing

Рис. 8: Log-likelihood vs. number of iterations

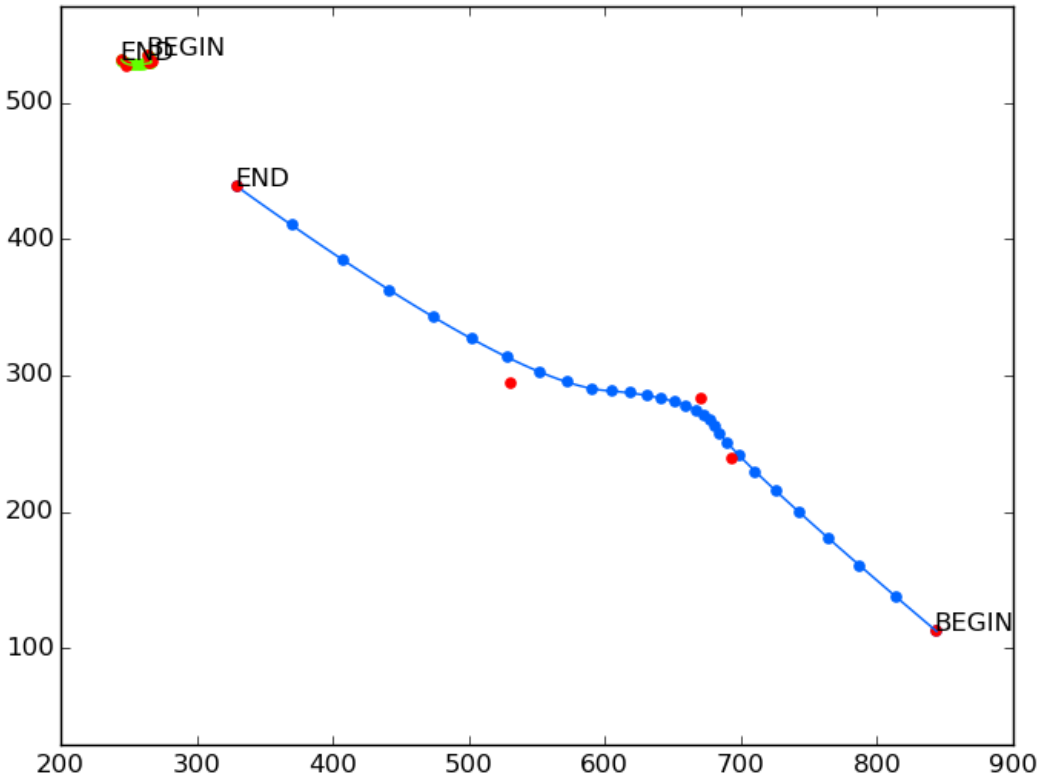
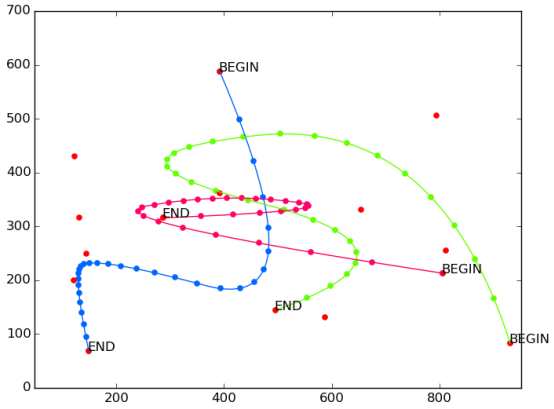
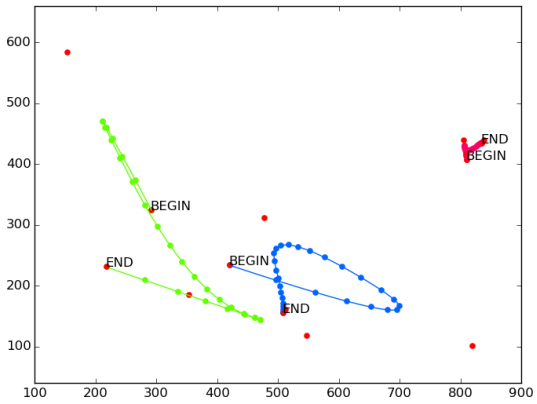


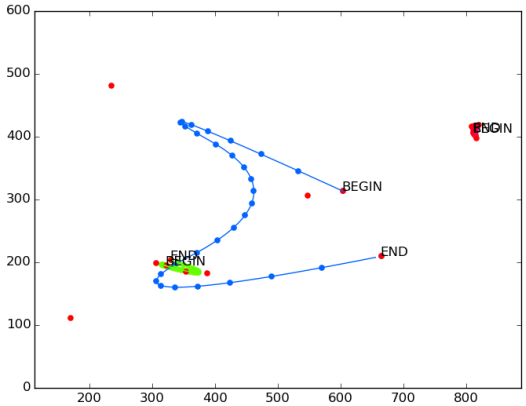
Рис. 9: Joint tracks for sentence «A person approached a stationary bicycle». Red points mean control points of splines. Points on line (one can see blue pints here) mean object position on frames extracted from the track. Blue line represents person's track and green – bicycle.



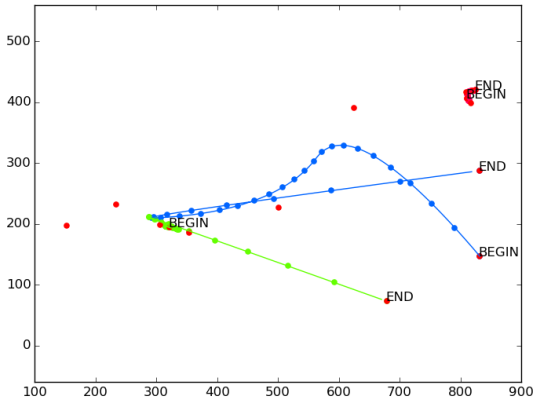
(a) index = 1, score = -2913



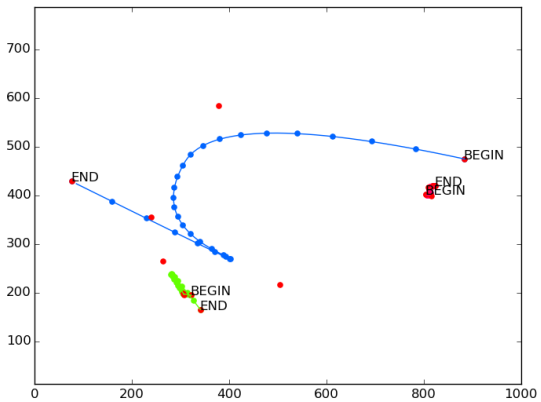
(b) index = 2, score = -2403



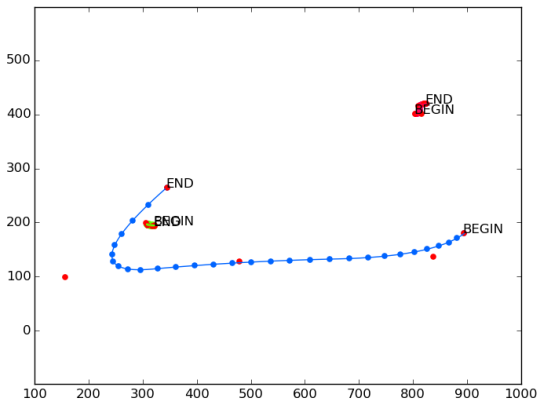
(c) index = 3, score = -2255



(d) index = 4, score = -2224



(e) index = 5, score = -2206



(f) index = 6, score = -2072

Рис. 10: Generated tracks for sentence «A person joined a stationary bicycle to the left of a stationary chair». Blue color means person, red means chair, green – bicycle. Red dots around track of the person and the bicycle are control points, while points on tracks are object positions on frames.

6. Question Answering

6.1. Parsing

Sentence parsing is performed by START [3]. First of all we need to extract all nouns from the text. Each of them will be an object in our system and we will generate a track for it. Secondly, we want to get all noun modifiers applied for those objects (such as «stationary»). Thirdly, we need to identify verbs and their modifiers. Every action word (verb) handles two objects: an agent and a patient. Adjective action modifiers are applied to the agent of action word.

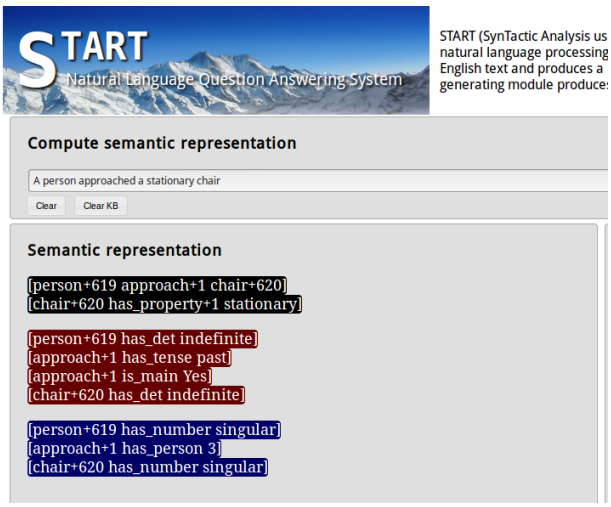


Рис. 11: START web interface

6.2. Tracks generation

Our next step is to generate tracks for all the objects. Here we slightly modify the conditions of Equation 9: we want to finish in the state of hyper word model that corresponds to the final states of all word models. This means that all actions are represented by words will be finalized. We also added the start and final empty states to make the models more general (to find a correct action on a video). One could see the difference in tracks with one word change on Figure 12.

To check how a (sentence, generated tracks) pair differs from other pairs, we conducted the following experiment: we create a set of similar sentences and calculate their score on tracks generated for a different sentence (for every sentence three tracks were generated). The result is presented in table 2. This experiment shows the following results:

Таблица 2: Sentences scores for generated tracks comparison. Sentences:

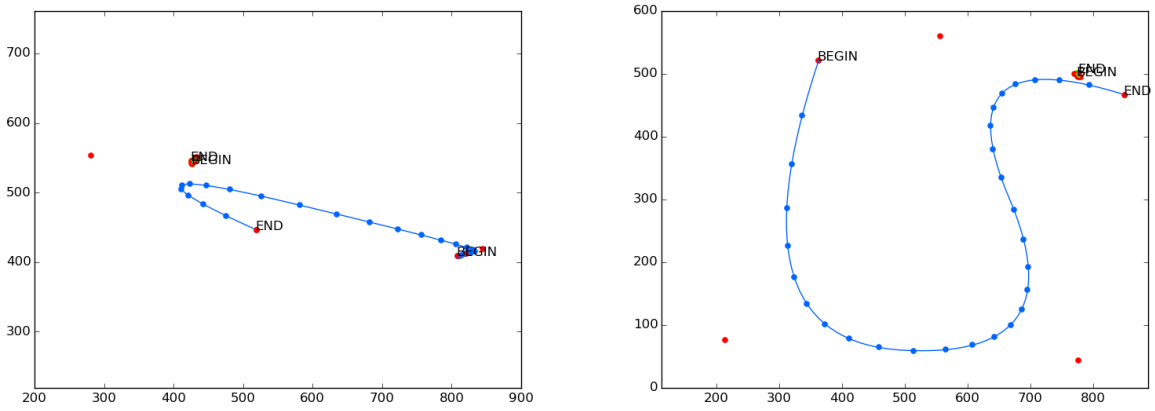
1. A person quickly approached a stationary bicycle
2. A person slowly approached a stationary bicycle
3. A person quickly joined a stationary bicycle
4. A bicycle quickly approached a stationary person
5. A bicycle slowly approached a stationary person
6. A person slowly joined a stationary bicycle

Tracks score for \ Tracks generated for	1	2	3	4	5	6
1	-1833.08	-1860.96	-1921.21	-2012.27	-2126.33	-2026.94
2	-1906.02	-1855.29	-1966.28	-2056.79	-2045.61	-1945.80
3	-2157.92	-2058.76	-2027.75	-2287.11	-2451.74	-2117.00
4	-2117.15	-2089.26	-2177.40	-1756.08	-1870.14	-2283.12
5	-2162.21	-2121.47	-2222.46	-1800.61	-1789.42	-2201.98
6	-2202.99	-2090.97	-2072.81	-2331.64	-2371.02	-2035.86

- Row r contains scores of pairs (sentence r, tracks for each sentence). As one can see, in every row the pair (sentence r, tracks for sentence r) is the one with the maximal score.

- Column c contains scores of pairs (a sentence, tracks for sentence c). Here we can see that for 4 out of 6 columns the score of the pair (sentence c, tracks for sentence c) is maximal in that column. The only exceptions are pairs 3-1 and 6-2, and there is an explanation: the model for «approach» gives a good score only to the tracks depicting horizontal movements (the «velocity orientation» distribution is 0.5 for moving to the left and 0.5 for moving to the right), whereas the model for «join» has a distribution of (0.25 for each direction in «top», «bottom», «left», «right»). So in case where «join» generates a horizontal track for the first object, sentence with a change of «join» $\rightarrow$ «approach» has a better score.

- Sentences with different participants but the same structure have different scores on



(a) A person slowly joined a stationary chair (b) A person quickly joined a stationary chair

Рис. 12: Difference in tracks for sentences with one different word. «Slowly» modifier means that score function stimulates tracks with big distance between adjacent frames, whereas «quickly» modifier encourages the sampling algorithm to expand the distance between object positions on different frames.

each other's tracks, that means that the class feature captures the difference well.

- These sentences have the same structure and the system tells them apart judging by their tracks. Sentences where a change («join» → «pick up») was applied differ much more significantly.

6.3. Answer parsing

Due to a relatively small number of types of questions that could be asked about object positions, we use pattern matching approach (from Section 1).

We asked the following questions:

- Who is near *obj* in time t?
- How far is obj_1 from obj_2 in time t?
- Where was *obj* in time t?
- Is obj_1 left of the obj_2 in time t?

- Is obj_1 right of the obj_2 in time t?

The first way to solve this task is to:

1. Generate tracks for input text
2. Generate possible answers
3. Rank answers using Equation 9
4. Choose the best answer

To improve this algorithm we could generate 3-4 different scenes (collections of tracks) and use them all. In this case we could use cosine similarity or correlation between answers scores vector and initial text scores vector for ranking. But this approach does not achieve appropriate quality with complex sentences. Also, this cannot work with questions about exact time.

Therefore, we use another way:

1. Generate 3-4 video-like media
2. Parse question using pattern approach and produce answers
3. Use absolute positions of objects in video to find the best answer

7. Other Applications

Following [17], we can use Section 4 results for annotating videos. This approach might also be useful for the following purposes:

1. Action detection on surveillance videos: recognition of potentially dangerous situations.

2. Automatic sports commentary generation

The second application of the proposed method is automatic map creation. Suppose you have a robot vacuum cleaner in a big house. You could inform your robot that your child is moving from room one to a chair in room two. The cleaner will reconstruct his movements to try to avoid him.

The third application is another machine translation approach. We transform sentence into interlingua visual information. Then we find the sentence in another language that fits best and return it as translation result.

8. Conclusion

This paper solves the problem of answering questions about movements of objects by generating video-like representation depicting these movements. We provide a new approach for answering spatial questions using generation of video-like information similar to human thinking. We propose two algorithms for object tracks generation (the objects are taken from the input sentence). Also, two algorithms for answers ranking are proposed (comparing sentences and direct video analysis). We achieved the following results: 74% questions about two input objects were answered correctly. Also, 52% of questions about sentences with three objects received a right answer. In the future this system could become a part of a larger QA system (and produce additional information for some spatial questions).

Future work suggestions are to try other types of generation algorithms, other types of tracks (instead of B-splines) and to speed up the algorithm (GPU realization).

Список литературы

[1] Dwivedi S. K., Singh V. Research and reviews in question answering system // International Conference on Computational Intelligence: Modeling Techniques and Applications. 2013.

[2] Green B., Wolf A., Chomsky C., Laughery K. Baseball: An automatic question answerer // Western Computing Conference. Vol. 19. 1961. P. 219–224.

[3] Katz B. Annotating the World Wide Web Using Natural Language // Proceedings of the 5th RIAO Conference on Computer Assisted Information Searching on the Internet. 1997.

[4] Katz B., Borchardt G., Felshin S. Natural Language Annotations for Question Answering // Proceedings of the 19th International FLAIRS Conference. 2006.

[5] Chung H., Song Y., Han K. et al. A Practical QA System in Restricted Domains. // 42nd Annual Meeting of the Association for Computational Linguistics (ACL). 2004.

[6] Mishra A., Mishra N., Agrawal A. Context-aware restricted geographical domain question answering system // IEEE International Conference on Computational Intelligence and Communication Networks (CICN). 2010. P. 548–553.

[7] Ittycheriah A., Zhu M. F. W., Ratnaparkhi A., Mammone R. IBM's statistical question answering system // Text Retrieval Conference TREC-9. 2000.

[8] Moschitti A. Answer filtering via text categorization in question answering systems. // 15 th IEEE International Conference on Tools with Artificial Intelligence. 2003. P. 241–248.

[9] Berger A., Caruana R., Cohn D. et al. Bridging the lexical chasm: statistical approaches to answer-finding // 23rd annual international ACM SIGIR conference on Research and development in information retrieval. 2000. P. 192– 199.

[10] Nie L., Wang M., Zha Z., Chua G. L. T.-S. Multimedia answering: enriching text QA with media information // 34th international ACM SIGIR conference on Research and development in Information Retrieval. 2011.

[11] Hong R., Tang J., Tan H.-K. et al. Beyond search: Event-driven summarization for web videos // ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM. 2011. Vol. 7.

[12] Gao L., Guangda, Zheng Y.-T. et al. Video Reference: A Video Question Answering Engine // 17th International ACM Conference in Multimedia. 2009.

[13] Hong R., Wang M., Li G. et al. Multimedia Question Answering // IEEE Multimedia. 2012.

[14] Katz B., Lin J., Stauffer C., Grimson E. Answering Questions about Moving Objects in Surveillance Videos // AAAI Technical Report SS-03-07. 2003.

[15] Malinowski M., Rohrbach M., Fritz M. Ask Your Neurons: A Neural-based Approach to Answering Questions about Images. arXiv. 2015. URL: `arXiv:1505.01121 [cs.CV]`.

[16] Yamoto J., Ohya J., Ishii K. Recognizing human action in time-sequential im- ages using hidden Markov model // IEEE Conference on Computer Vision and Pattern Recognition. 1992. P. 379–385.

[17] Yu H., Siddharth N., Barbu A., Siskind J. M. A Compositional Framework for Grounding Language Inference, Generation, and Acquisition in Video // Journal of Artificial Intelligence Research. 2015. — 04. Vol. 52. P. 601–713.

[18] Viterbi A. J. Error bounds for convolutional codes and an asymtotically optimum decoding algorithm // IEEE Transactions on Information Theory. 1967.

[19] Starner T., Weaver J., Pentland A. Real-time American Sign Language recognition using desk and wearable computer based video // IEEE Transactions on Pattern Analysis and Machine Intelligence. 1998. Vol. 20, no. 12. P. 1371–1375.

[20] Girshick R. B., Felzenszwalb P. F., McAllester D. Discriminatively Trained Deformable Part Models, Release 5. http://people.cs.uchicago.edu/ rbg/latent-release5/.

[21] Felzenszwalb P. F., Girshick R. B., McAllester D., Ramanan D. Object Detection with Discriminatively Trained Part Based Models // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2010. Vol. 32, no. 9. P. 1627–1645.

[22] Horn B., Schunck B. Determining Optical Flow // Artificial Intelligence. 1981.

[23] Bradski G. The OpenCV Library // Dr. Dobb's Journal of Software Tools. 2000.

[24] Farneback G. Two-frame Motion Estimation Based on Polynomial Expansion // Proceedings of the 13th Scandinavian Conference on Image Analysis. SCIA'03. Berlin, Heidelberg: Springer-Verlag, 2003. P. 363–370. URL: http://dl.acm.org/citation.cfm?id=1763974.1764031.

[25] Hastings W. K. Monte Carlo Sampling Methods Using Markov Chains and Their Applications // Biometrika. 1970. — 04. Vol. 57, no. 1. P. 97–109.

[26] Kirkpatrick S., Gelatt C. D., Vecchi M. P. Optimization by Simulated Annealing S. Kirkpatrick; C. D. Gelatt; M. P. Vec // Science. 1983. — May. Vol. 220,, no. 4598. P. 671–680.